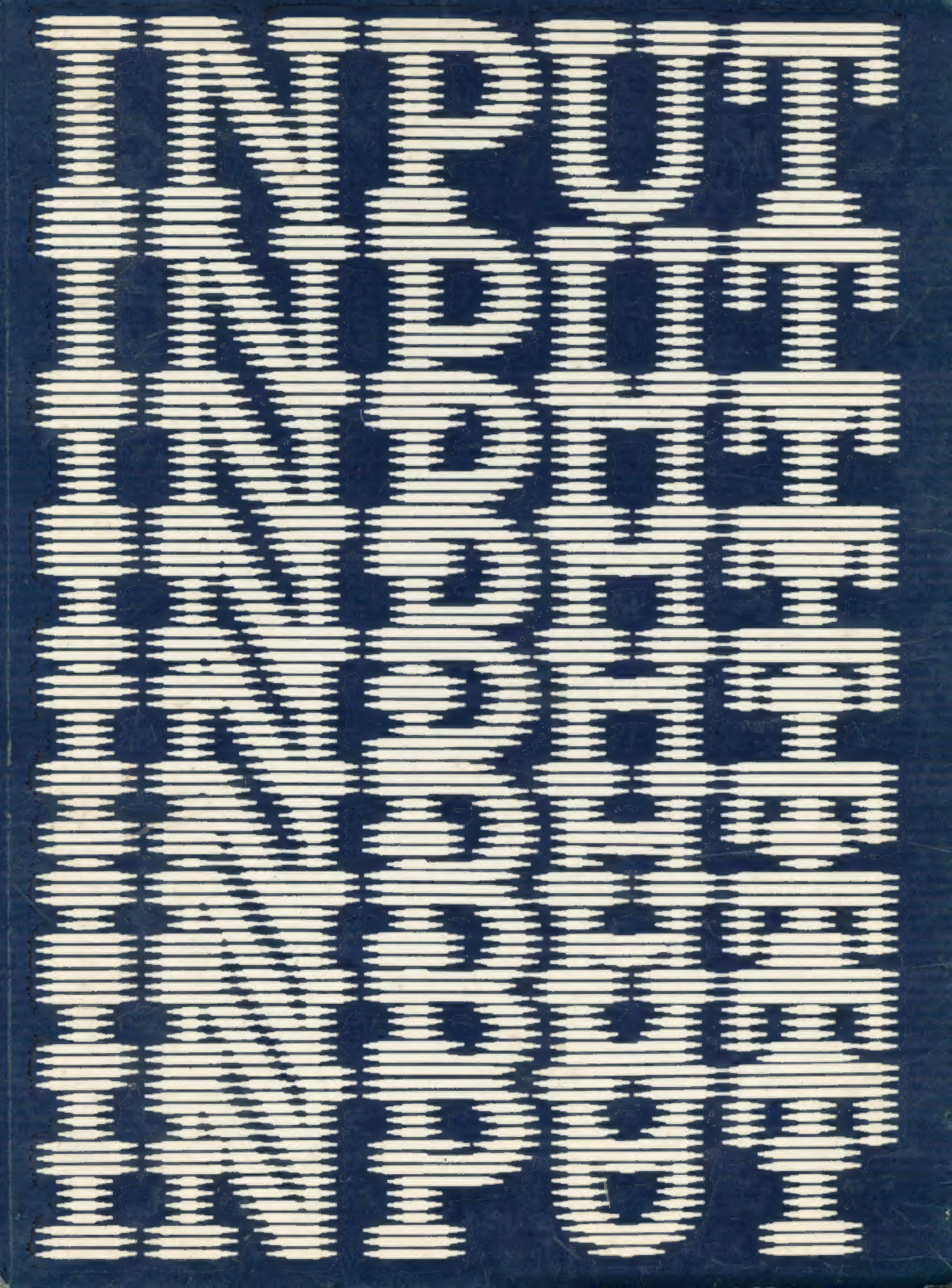

INPUT
INPUT
INPUT
INPUT
INPUT
INPUT
INPUT
INPUT
INPUT

INPUT

3

Editor
VICTOR CIVITA

REDAÇÃO

Diretora Editorial: Iara Rodrigues

Editor Executivo: Antonio José Filho

Editor Chefe: Paulo de Almeida
Editor de Texto: Cláudio A. V. Cavalcanti
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Grace Alonso Arruda,
Monica Lenardon Corradi
Secretária de Redação/Coordenadora: Stefania Crema
Secretários de Redação: Beatriz Hagström,
José Benedito de Oliveira Damião, Maria de Lourdes
Carvalho, Marisa Soares de Andrade, Mauro de Queiroz

COLABORADORES
Consultor Editorial Responsável: Dr. Renato M. E. Sabbatini
(Diretor do Núcleo de Informática Biomédica
da Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria
em Informática Ltda., Campinas, SP

Tradução: Reinaldo Cúrcio

Tradução, adaptação, programação e redação:
Abílio Pedro Neto, Aluísio J. Dornellas de Barros,
Marcelo R. Pires Therezo, Raul Neder Porrelli
Coordenação Geral: Rejane Felizatti Sabbatini
Editora de Texto: Ana Lúcia B. de Lucena
Assistente de Arte: Dagmar Bastos Sampaio

COMERCIAL
Diretor Comercial: Roberto Martins Silveira
Gerente Comercial: Flávio Maculan
Gerente de Circulação: Denise Maria Mozol

PRODUÇÃO
Gerente de Produção: João Stungis
Coordenador de Impressão: Atílio Roberto Bonon
Preparador de Texto/Coordenador: Eliel Silveira Cunha
Preparadores de Texto: Alzira Moreira Braz,
Ana Maria Dilguerian, Karina Ap. V. Grechi,
Levon Yacubian, Luciano Tasca, Maria Teresa Galluzzi,
Maria Teresa Martins Lopes, Paulo Felipe Mendrone
Revisor/Coordenador: José Maria de Assis
Revisoras: Conceição Aparecida Gabriel,
Isabel Leite de Camargo, Ligia Aparecida Ricetto,
Maria de Fátima Cardoso, Nair Lucia de Britto
Paste-up: Anastase Potaris, Balduino F. Leite, Edson Donato

Edição organizada pela Editora Nova Cultural Ltda.
Av. Brigadeiro Faria Lima, n.º 2000 - 3.º andar
CEP 01452 - São Paulo - SP - Brasil
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta na AM Produções Gráficas Ltda.
e impressa na Divisão Gráfica da Editora Abril S.A.

SUMÁRIO

APLICAÇÕES

CÓDIGO DE MÁQUINA

PERIFÉRICOS

PROGRAMAÇÃO BASIC

PROGRAMAÇÃO DE JOGOS

UM SIMULADOR DE VÔO (2)

- UM PILOTO AUTOMÁTICO PARA CONDUZIR O AVIÃO
- APROXIME-SE DA PISTA
- COMO TRAÇAR O RUMO
- FAÇA O PAINEL FUNCIONAR



Ligue as turbinas do avião e observe o jogo dos ponteiros no painel dos mostradores. Mas não deixe de apertar o cinto, pois o piloto automático é meio maluco.

Já vimos como reproduzir o interior de uma cabine de avião na tela do microcomputador. No programa do artigo anterior para as linhas TRS-Color, MSX, Apple e TK-2000, o avião foi deixado estático, em pleno ar. Evidentemente, uma situação assim nunca ocorre na vida real.

Neste artigo, nosso aeroplano vai sair voando e os instrumentos do painel vão ganhar vida, para que possamos acompanhar seu movimento, apesar de ainda não estarmos em condições de mantê-lo sob controle.

COMO FAZER O AEROPLANO VOAR

Esta é certamente a parte mais longa da listagem: uma série complexa de variáveis independentes deve ser constantemente calculada para reproduzir e controlar o comportamento do aparelho em vôo. Ao mesmo tempo, os instrumentos do painel precisam ser redesenhados à medida que a posição e a altitude do avião sofrem modificações. Da mesma forma, os números dos mostradores são impressos para que saibamos sua posição e sua orientação.

APROXIME-SE DA PISTA

Uma imagem da pista no radar nos mostra o ângulo de aproximação no Spectrum e no MSX. No TRS-Color, uma imagem da pista pode ser observada através da janela, à medida que chegamos próximo à pista. Esta aparece progressivamente, aproveitando a capacidade do TRS-Color em desenhar elipses. Já os usuários do Apple e do TK-2000 devem se orientar pela bússola (*bearing*), pela direção da pista (*runway*) e pela distância do avião em relação ao eixo longitudinal da pista (*drift*).

O CURSO DO APARELHO

Diversos fatores devem ser considerados para que se possa calcular a posição do aparelho num dado momento. A direção em que estamos seguindo, por exemplo, é afetada pela dos ventos e pela posição dos ailerons instalados nas duas asas. A velocidade de cruzeiro depende, por sua vez, da intensidade do vento. As velocidades de ascensão e de mergulho estão sempre relacionadas

com a de cruzeiro, e assim por diante.

Para manter sempre atualizados os valores nos mostradores, as variáveis têm de ser recalculadas e colocadas novamente na tela.

S

```
2 LET WY=0: LET WX=0: LET GZ=0: LET GY=0: LET GX=0
5 LET RW=0: LET Y1=120: LET Y2=120: LET Y3=40: LET Y4=40: LET POW=0: LET GC=0: LET RB=0: LET LL=0: LET YC=0: LET AD=0: LET ST=0: LET RL=0: LET BC=0: LET NC=0: LET PT=0: LET PX=0: LET VZ=0: LET VY=0: LET VX=0
500 LET RA=AD*C: LET VX=AS*SIN RA
510 LET VY=AS*COS RA: RETURN
1000 LET PZ=PZ+GZ: LET PY=PY+GY: LET PX=PX+GX
1025 IF ST=1 THEN PRINT OVER 1;AT 4,9;"M E R G U L H O": LET ST=0: GOTO 1040
1030 IF AS<30 THEN GOSUB 1500
1040 LET AD=AD+RL: IF AD<0 THEN LET AD=AD+360
1050 IF AD>359 THEN LET AD=AD-360
1060 LET VZ=AS*SIN (PT*C)-10+AS/15
1070 LET GZ=VZ: LET GY=VY+WY: LET GX=VX+WX
1080 IF VY=0 THEN LET GD=-PI/2: GOTO 1100
1090 LET GD=-ATN (VX/VY)/C
1100 GOSUB 500
1110 RETURN
1500 LET ST=1: PRINT OVER 1;AT 4,9;"M E R G U L H O": FOR M=1 TO 4: FOR N=20 TO -20 STEP -4: SOUND .01,N: NEXT N: NEXT M
1510 LET RL=INT (RND*21)-9: LET PT=-21-INT (RND*5)
1520 RETURN
2180 IF GC<>0 THEN GOSUB 2200
2190 LET AS=AS+16*(TC*30-AS-8*PT)/AS: GOSUB 2200: GOTO 2205
2200 PLOT 35,50: DRAW OVER 1;15*SIN (AS*PI/200),15*COS (AS*PI/200): RETURN
2205 IF GC<>0 THEN PLOT 155,50: DRAW OVER 1;10*SIN (TN*PI/5),10*COS (TN*PI/5): PLOT 155,50: DRAW OVER 1;15*SIN (UN*PI/500),15*COS (UN*PI/500)
2210 LET TN=PZ/1000: LET UN=PZ-1000*INT TN: PLOT 155,50: DRAW OVER 1;10*SIN (TN*PI/5),10*COS (TN*PI/5): PLOT 155,50: DRAW OVER 1;15*SIN (UN*PI/500),15*COS (UN*PI/500)
2220 IF GC<>0 THEN GOSUB 2230
2225 IF POW=-1 AND TC>2 THEN LET TC=TC-2
2226 IF POW=1 AND TC<8.8 THEN LET TC=TC+2
2228 GOSUB 2230: GOTO 2240
2230 PLOT 215,50: DRAW OVER 1;15*SIN (TC*PI/5),15*COS (TC*PI/5): RETURN
2240 PRINT AT 21,2;ABS INT AD;" "
2250 IF PY=0 THEN LET RB=0: GOTO 2260
2255 LET RB=ATN (PX/PY)/C: IF PY>0 THEN LET RB=RB+180
2260 IF RB<0 THEN LET RB=RB+360
2270 PRINT AT 21,10;INT RB;" ";AT 21,18;ABS INT PX;" "
2280 PRINT AT 21,25;INT (SQR (PY*PY+PX*PX));" "
2290 IF (Y1<=110 AND Y2<=110) OR (Y1>=130 AND Y2>=130) THEN GOTO 2300
2295 IF GC<>0 THEN PLOT OVER 1;X1,168-Y1: DRAW OVER 1;X2-PEEK 23677,168-Y2-PEEK 23678
2300 LET YC=120+(PT/3): LET X1=80: LET X2=110: LET Y1=YC+17*TAN (RL*2*C): LET Y2=YC-17*TAN (RL*2*C)
2310 IF (YC<110 OR YC>130) AND RL=0 THEN GOTO 2376
2320 IF Y1<110 THEN LET X1=95-(95-X1)*(110-YC)/(Y1-YC): LET Y1=110: GOTO 2340
2330 IF Y1>130 THEN LET X1=95-(95-X1)*(130-YC)/(Y1-YC): LET Y1=130
2340 IF Y2<110 THEN LET X2=95-(95-X2)*(110-YC)/(Y2-YC): LET Y2=110: GOTO 2360
2350 IF Y2>130 THEN LET X2=95-(95-X2)*(130-YC)/(Y2-YC): LET Y2=130
2360 IF X1<80 OR X2>110 THEN GOTO 2376
2370 PLOT OVER 1;X1,168-Y1: DRAW OVER 1;X2-PEEK 23677,168-Y2-PEEK 23678
2376 IF (RL=RR AND PP=PT) THEN GOTO 2500
2377 IF (Y3<=2 AND Y4<=2) OR (Y3>=90 AND Y4>=90) THEN GOTO 2380
2378 IF GC<>0 THEN PLOT OVER 1;X3,176-Y3: DRAW OVER 1;X4-PEEK 23677,(176-Y4)-PEEK 23678
2380 LET YC=33+PT*4: LET X3=11: LET X4=244: LET Y3=YC+118*TAN (RL*2*C): LET Y4=YC-118*TAN (RL*2*C)
2390 IF (YC<2 OR YC>90) AND RL=0 THEN GOTO 2450
2400 IF Y3<2 THEN LET X3=128-(128-X3)*(2-YC)/(Y3-YC): LET Y3=2: GOTO 2420
2410 IF Y3>90 THEN LET X3=128-(128-X3)*(90-YC)/(Y3-YC): LET Y3=90
2420 IF Y4<2 THEN LET X4=128-(128-X4)*(2-YC)/(Y4-YC): LET Y4=2: GOTO 2440
2430 IF Y4>90 THEN LET X4=128-(128-X4)*(90-YC)/(Y4-YC): LET Y4=90
2440 IF X3<11 OR X4>244 THEN GOTO 2500
2445 OVER 1: PLOT X3,176-Y3: DRAW X4-PEEK 23677,(176-Y4)-PEEK 23678: OVER 0
2500 GOSUB 8000
2505 IF GC=0 THEN LET GC=1
2510 LET RR=RL: LET PP=PT: RETURN
5080 LET GZ=VZ: LET GY=VY+WY: LET GX=VX+WX
5090 LET TC=5
5100 LET RT=3: LET TP=5: LET WR=50
5500 IF INT (RND*5)=1 THEN LET RL=RL+INT (RND*5)-2: IF INT (RND*5)=1 THEN LET PT=PT+3-INT (RND*2)+1*2
5510 GOSUB 1000: IF PZ<0 THEN GOTO 5530
5520 GOSUB 2180: GOTO 5500
5530 GOTO 5500
8000 IF GC<>0 THEN PLOT 127,174: DRAW OVER 1;OX,OY
8010 LET OX=16*SIN (RB*(PI/180)): LET OY=-(16*ABS COS (RB*(PI/180)))
8020 PLOT 127,174: DRAW OVER 1;OX,OY
8025 LET WB=AD: IF AD>180 THEN LET WB=WB-360
8026 IF RB>180 THEN LET WB=WB+360-RB: GOTO 8040
8030 LET WB=WB-RB
8040 IF RW=1 THEN PLOT OVER 1;RDX,175-RDY
8050 LET RW=0: IF ABS WB>57 THEN RETURN
8060 LET RDX=X3+INT (((X4-X3)/2)-SIN (WB*(PI/180))*(X4-X3)*6)
8070 LET RDY=Y3+((Y4-Y3)*((RDX-X3)/(X4-X3))+2)
8080 IF RDY<2 OR RDY>90 OR RDX<11 OR RDX>244 THEN RETURN
```



```
8090 LET RW=1: PLOT  OVER 1;RDX
,175-RDY
8100 RETURN
```

O **POKE** na linha 1 trava as linhas maiúsculas do computador. A linha 5 zera todas as variáveis. A linha 110, que já foi digitada na primeira parte do programa (ou seja, no artigo anterior), envia o computador para a linha 5000, onde é desenhado o interior da cabine. A linha 5080 cuida das variáveis que determinam a posição do avião no espaço: **GZ** se refere à distância do aparelho ao longo do eixo **Z** — sua altitude; **VZ** é a velocidade ao longo desse mesmo eixo; **VY** é a velocidade ao longo do eixo **Y** — para frente e para trás; **WY** é a velocidade do vento nessa mesma direção. **GX**, **VX** e **WX** correspondem à distância percorrida, velocidade e velocidade do vento em relação ao eixo **X** — direita e esquerda.

A linha 5090 estabelece o valor inicial da rotação do motor. A linha 5100 define os limites da inclinação do avião (**RT**, relacionada com os movimentos laterais), assim como da ascensão (**TP**, que faz o avião subir ou descer) e da largura da pista (**WR**).

A linha 5500 é um comando temporário encarregado de manobrar o avião até o momento em que a rotina de controle propriamente dita é digitada (o que será feito no próximo artigo desta série).

A linha 5510 chama a sub-rotina que começa na linha 1000 e termina em 1110. Essa sub-rotina recalcula os valores de todas as variáveis à medida que o avião se movimenta. As linhas 1025 e 1030 verificam se o aparelho entrou em parafuso, por ter uma velocidade muito baixa (inferior a 30 m por segundo). Caso isso aconteça, será chamada a sub-rotina 1500, que se encarrega de reproduzir os efeitos de um mergulho. A linha 5510 impede que possamos prever o rumo do avião durante a queda (este pode mergulhar diretamente ou então entrar em parafuso).

A linha 1100 chama a sub-rotina 500, que calcula o ângulo da trajetória do avião.

A próxima grande sub-rotina a ser chamada começa na linha 2180 e termina em 2510. Ela redesenha os mostradores e contadores à medida que a informação que estes trazem se modifica. A linha 2180 verifica o contador **GC** para ver se há algo a ser redesenhado. A sub-rotina 2200 desenha a nova posição dos ponteiros do mostrador da velocidade do vento (*airspeed*). As linhas 2205 e 2210 calculam e redesenham as duas novas posições dos ponteiros que marcam a altitude. A linha 2230 movimenta os ponteiros, usando os cálculos das linhas 2225 e 2226.

O contagiros é atualizado pela linha 2240. As linhas 2250 a 2270 calculam a direção da pista (*runway*) e o desvio do avião em relação ao eixo principal desta última (*drift*). A distância até o centro da pista é calculada e impressa pela linha 2280.

O horizonte artificial do segundo mostrador é calculado e desenhado pelas linhas 2280 a 2370.

A linha 2376 verifica o horizonte que aparece na janela, enquanto as linhas 2377 a 2445 o recalculam e o redesenham, conforme o caso.

A linha 2500 chama a sub-rotina que começa na linha 8000 e termina na 8100. Essa sub-rotina calcula e desenha a imagem da pista no radar, que pode ser vista no topo da tela, e o pontinho logo abaixo da linha de horizonte.



```
100 PRINT RND(-TIME)
500 RA=AD*C:VX=VV*SIN(RA)
510 VY=VV*COS(RA):RETURN
1000 PZ=PZ+GZ:PY=PY+GY:PX=PX+GX
1020 VV=VV+16*(TC*30-VV-8*PT)/V
V
1030 IF ST=1 THEN LINE(90,1)-(2
30,8),4,BF:ST=0:GOTO 1050
1040 IF VV<30 THEN GOSUB 1500
1050 AD=AD+RL:IF AD<0 THEN AD=A
D+360
1060 IF AD>359 THEN AD=AD-360
1070 VZ=VV*SIN(PT*C)-10+VV/15
1080 GZ=VZ:GY=VY+WY:GX=VX+WX
1090 IF VY=0 THEN GD=-PI/2:GOTO
 1110
1100 GD=-ATN(VX/VY)/C
1110 GOSUB 500
1120 RETURN
1500 ST=1:X=96:Y=1:COLOR 6
1510 A$="MERGULHO":GOSUB 4000:X
=97:GOSUB 4000
1520 PLAY"T255AGFEDCAGFEDC"
1530 RL=INT(RND(1)*21)-9:PT=-21
-INT(RND(1)*5)
1540 RETURN
2000 COLOR 15
2005 LINE(35,128)-(35+15*SIN(V1
*PI/200),128-15*COS(V1*PI/200))
,8
2010 LINE(35,128)-(35+15*SIN(VV
*PI/200),128-15*COS(VV*PI/200))
2015 LINE(155,128)-(155+10*SIN(
T1*PI/5),128-10*COS(T1*PI/5)),8
2020 TN=PZ/1000:UN=PZ-1000*INT(
TN):LINE(155,128)-(155+10*SIN(T
N*PI/5),128-10*COS(TN*PI/5))
2025 LINE(155,128)-(155+15*SIN(
U1*PI/500),128-15*COS(U1*PI/500
)),8
2030 LINE(155,128)-(155+15*SIN(
UN*PI/500),128-15*COS(UN*PI/500
))
2035 LINE(215,128)-(215+15*SIN(
T2*PI/5),128-15*COS(T2*PI/5)),8
2040 LINE(215,128)-(215+15*SIN(
TC*PI/5),128-15*COS(TC*PI/5))
2050 COLOR 15:X=16:Y=180:A$=STR
```



```
$(ABS(INT(AD))):GOSUB 3900
2060 IF PY=0 THEN RB=0 ELSE RB=
ATN(PX/PY)/C:IF PY>0 THEN RB=RB
+180
2070 IF RB<0 THEN RB=RB+360
2080 X=79:A$=STR$(INT(RB)):GOSU
B 3900:X=138:A$=STR$(INT(PX)):G
OSUB 3900
2090 X=194:A$=STR$(INT(SQR(PY*P
Y+PX*PX))):GOSUB 3900
2100 YC=128+PT:X1=80:X2=110:Y1=
YC+17*TAN(RL*2*C):Y2=YC-17*TAN(
RL*2*C)
2110 IF(YC<113 OR YC>143) AND R
L=0 THEN 2320
2120 IF Y1<113 THEN X1=95-(95-X
1)*(105-YC)/(Y1-YC):Y1=113:GOTO
 2140
2130 IF Y1>143 THEN X1=95-(95-X
1)*(135-YC)/(Y1-YC):Y1=143
2140 IF Y2<113 THEN X2=95-(95-X
2)*(105-YC)/(Y2-YC):Y2=113:GOTO
 2160
2150 IF Y2>143 THEN X2=95-(95-X
2)*(135-YC)/(Y2-YC):Y2=143
2160 IF X1<80 OR X2>110 THEN 21
80
2165 DRAW "BM81,128C10R9F5E5R9"
2170 LINE(X5,Y5)-(X6,Y6),8:LINE
(X1,Y1)-(X2,Y2):X5=X1:Y5=Y1:X6=
X2:Y6=Y2
2180 IF X7-R>10 AND X7+R<245 AN
D Y7>0 AND Y7<80 THEN CIRCLE(X7
,Y7),R,4,0,.5,10
2190 IF RL=RR AND PP=PT THEN 22
90
2200 IF HF=1 THEN LINE (X3,Y3)-
(X4,Y4),4
2210 HF=0:YC=33+PT*4:X3=16:X4=2
39:Y3=YC+118*TAN(RL*2*C):Y4=YC-
118*TAN(RL*2*C)
2220 IF(YC<10 OR YC>79) AND RL=
0 THEN 2290
2230 IF Y3<1 THEN X3=128-(128-X
3)*(1-YC)/(Y3-YC):Y3=1:GOTO 225
0
2240 IF Y3>79 THEN X3=128-(128-
X3)*(79-YC)/(Y3-YC):Y3=79
2250 IF Y4<1 THEN X4=128-(128-X
4)*(1-YC)/(Y4-YC):Y4=1:GOTO 227
0
2260 IF Y4>79 THEN X4=128-(128-
X4)*(79-YC)/(Y4-YC):Y4=79
2270 IF X3<16 OR X4>239 THEN 22
90
2280 HF=1:LINE(X3,Y3)-(X4,Y4),1
5
2290 WB=AD:IF AD>180 THEN WB=WB
-360
2300 IF RB>180 THEN WB=WB+360-R
B ELSE WB=WB-RB
2310 IF ABS(WB)>60 AND ABS(PY)>
1000 THEN 2370
2320 AN=118/(60*SQR((X3-X4)*(X3
-X4)+(Y3-Y4)*(Y3-Y4))):X7=(X3+X
4)/2+SGN(X3-X4)+WB*AN*(X3-X4)
2330 Y7=(Y3+Y4)/2+2+WB*AN*(Y3-Y
4):IF X7<16 OR X7>239 OR Y7<1 O
R Y7>79 THEN 2370
2340 IF ABS(PY)<1000 THEN R=8-Y
7/10 ELSE R=4000/ABS(PY):IF R*1
0+Y7>80 THEN R=8-Y7/10
2350 IF Y7<1 OR Y7>79 OR X7-R<1
6 OR X7+R>239 THEN 2370
2360 CIRCLE(X7,Y7),R,15,0,.5,10
2370 V1=VV:U1=UN:T1=TN:T2=TC
2380 RR=RL:PP=PT:RETURN
3900 LINE(X,Y)-(X+50,Y+8),2,BF
5080 GZ=VZ:GY=VY+WY:GX=VX+WX
5090 TC=5
5100 RT=3:TP=5:WR=50
5190 DRAW "BM81,128C10R9F5E5R9"
5500 IF INT(RND(1)*10)=1 THEN R
L=RL-SGN(RL)+INT(RND(1)*5-2)
5505 IF INT(RND(1)*10)=1 THEN P
T=PT+3-INT(RND(1)*2+1)*2
5510 GOSUB 1000:IF PZ<0 THEN 55
30
5520 GOSUB 2000:GOTO 5500
5530 GOTO 5530
```

A linha 100 "embaralha as cartas", permitindo a criação de números ao acaso. A 5080 estabelece os valores das variáveis que controlam a posição do aparelho no espaço, conforme a sua velocidade e a força e direção do vento. **GZ** nos revela a distância percorrida ao longo do eixo Z (alto-baixo); **VZ**, a velocidade nesse eixo. **VY** informa a velocidade ao longo do eixo **Y** (frente-trás); **WY**, a velocidade do vento nessa mesma direção; **GX**, **VX** e **WX** correspondem respectivamente à distância percorrida, à velocidade do aparelho e à velocidade do vento ao longo do eixo **X** (direita-esquerda).

A linha 5090 determina a rotação inicial do motor; a 5100 define os limites da inclinação lateral (**RT**), que faz o avião virar; da inclinação frontal (**TP**), que faz o avião subir ou descer, e da largura da pista (**WR**).

As últimas cinco linhas são comandos temporários. Elas fazem com que o avião voe de maneira irregular, com inclinações laterais e frontais estabelecidas ao acaso.

As sub-rotinas das linhas 1000 a 1120 recalculam todas as variáveis enquanto o aeroplano se movimenta. As linhas 1030 e 1040 verificam se a velocidade caiu abaixo de 30 m por segundo. Quando isso acontece, é chamada a sub-rotina da linha 1500, que recria os efeitos de uma queda ou mergulho. A linha 1510 introduz um elemento de acaso no processo, de maneira que tanto podemos mergulhar diretamente para o chão como podemos entrar em parafuso. A sub-rotina 500, que é chamada pela linha 1110, calcula o ângulo em que o aparelho está voando.

A sub-rotina que vai da linha 2000 à linha 2380 cuida do restante do programa. Ela atualiza a tela, ajustando o painel de instrumentos, que mostra o movimento do avião. Para fazer isso, é necessário, primeiramente, apagar o antigo conteúdo de cada instrumento e desenhar a seguir os ponteiros correspondentes à nova posição do aparelho.

O mostrador da velocidade do vento é atualizado pelas linhas 2005 e 2010, enquanto as linhas 2015, 2020, 2025 e 2030 cuidam do mostrador de altitude. O contagiros é modificado pelas linhas 2035 e 2040; a linha 2050 corrige a direção em que o avião está seguindo (*bearing*, que é o ângulo da bússola medido em graus). A direção da pista (*runway*, também em graus) é modificada pelas linhas 2060 a 2080, e o novo valor da distância é impresso pela linha 2090.

A linha 2190 verifica se o horizonte deve ser modificado, enquanto as linhas 2100 a 2170 desenham o novo horizonte artificial no mostrador correspondente. O horizonte é desenhado pelas linhas 2200 a 2280.

A nova posição da pista é calculada pelas linhas 2290 a 2360. A 2350 verifica se a pista pode ser desenhada na tela. A 2180 apaga a versão antiga quando uma nova é desenhada.

Execute então o programa e veja o avião voar sozinho... por enquanto.



```
800 RA = AD * C:VX = AS *  SIN
(RA)
810 VY = AS *  COS (RA): RETURN

1000 PZ = PZ + GZ:PY = PY + GY:
PX = PX + GX
1020 AS = AS + 16 * (TC * 30 -
AS - 8 * PT) / AS
1030  IF ST = 1 THEN ST = 0: GO
TO 1050
1040  IF AS < 30 THEN  GOSUB 15
00
1050 AD = AD + RL: IF AD < 0 TH
EN AD = AD + 360
1060  IF AD > 359 THEN AD = AD
- 360
1070 VZ = AS *  SIN (PT * C) -
10 + AS / 15
1080 GZ = VZ:GY = VY + WY:GX =
VX + WX
1090  IF VY = 0 THEN GD =  - PI
 / 2: GOTO 1110
1100 GD =  -  ATN (VX / VY) / C
1110  GOSUB 800
1120  RETURN
1500 ST = 1
1510 RL =  INT ( RND (1) * 21 +
 1) - 10:PT =  - 20 -  INT ( RN
D (1) * 5 + 1)
1520  RETURN
2000  HCOLOR= 0
2010  HPLOT 41,134 TO 41 + 20 *
  SIN (A1 * PI / 200),134 - 20
*  COS (A1 * PI / 200)
2015  HCOLOR= 3: HPLOT 41,134 T
O 41 + 20 *  SIN (AS * PI / 200
),134 - 20 *  COS (AS * PI / 20
0)
2020  HCOLOR= 0: HPLOT 177,134
TO 177 + 15 *  SIN (T1 * PI / 5
```

```
),134 - 15 *  COS (T1 * PI / 5)
2025  HCOLOR= 3:TN = PZ / 1000:
UN = PZ - 1000 *  INT (TN): HPL
OT 177,134 TO 177 + 15 *  SIN (
TN * PI / 5),134 - 15 *  COS (T
N * PI / 5)
2030  HCOLOR= 0: HPLOT 177,134
TO 177 + 20 *  SIN (U1 * PI / 5
00),134 - 20 *  COS (U1 * PI /
500)
2035  HCOLOR= 3: HPLOT 177,134
TO 177 + 20 *  SIN (UN * PI / 5
00),134 - 20 *  COS (UN * PI /
500)
2040  HCOLOR= 0: HPLOT 245,134
TO 245 + 20 *  SIN (T2 * PI / 5
),134 - 20 *  COS (T2 * PI / 5)
2045  HCOLOR= 3: HPLOT 245,134
TO 245 + 20 *  SIN (TC * PI / 5
),134 - 20 *  COS (TC * PI / 5)
2050 L = 23:C = 4:A$ =  LEFT$ (
 STR$ ( ABS ( INT (AD))) + "
",3): GOSUB 4000
2060  IF PY = 0 THEN RB = 0: GO
TO 2070
2062 RB =  ATN (PX / PY) / C
2065  IF PY > 0 THEN RB = RB +
180
2070  IF RB < 0 THEN RB = RB +
360
2080 C = 14:A$ =  LEFT$ ( STR$
( INT (RB)) + "   ",3): GOSUB 4
000:C = 24:A$ =  LEFT$ ( STR$ (
 INT (PX)) + "    ",4): GOSUB 4
000
2090 C = 32:A$ =  LEFT$ ( STR$
( INT ( SQR (PY * PY + PX * PX)
)) + "     ",5): GOSUB 4000
2100 YC = 134 + PT:X1 = 90:X2 =
 130:Y1 = YC + 17 *  TAN (RL *
2 * C):Y2 = YC - 17 *  TAN (RL
* 2 * C)
2110  IF (YC < 119 OR YC > 149)
 AND RL = 0 THEN 2380
2120  IF Y1 < 119 THEN X1 = 110
 - (110 - X1) * (119 - YC) / (Y
1 - YC):Y1 = 119: GOTO 2140
2130  IF Y1 > 149 THEN X1 = 110
 - (110 - X1) * (149 - YC) / (Y
1 - YC):Y1 = 149
2140  IF Y2 < 119 THEN X2 = 110
 - (110 - X2) * (119 - YC) / (Y
2 - YC):Y2 = 119
2150  IF Y2 > 149 THEN X2 = 110
 - (110 - X2) * (149 - YC) / (Y
2 - YC):Y2 = 149
2160  IF X1 < 90 OR X2 > 130 TH
EN 2190
2170  HCOLOR= 0: HPLOT X5,Y5 TO
 X6,Y6
2175  HCOLOR= 3: HPLOT X1,Y1 TO
 X2,Y2
2177  HCOLOR= 5: HPLOT 85,138 T
O 104,138 TO 109,148 TO 114,138
 TO 134,138
2190  IF RL = RR AND PP = PT TH
EN 2380
2200  IF HF = 1 THEN  HCOLOR= 0
: HPLOT X3,Y3 TO X4,Y4
2210 HF = 0:YC = 33 + PT * 4:X3
 = 11:X4 = 275:Y3 = YC + 118 *
 TAN (RL * 2 * C):Y4 = YC - 118
 *  TAN (RL * 2 * C)
2220  IF (YC < 10 OR YC > 79) A
ND RL = 0 THEN 2380
2230  IF Y3 < 1 THEN X3 = 143 -
 (143 - X3) * (1 - YC) / (Y3 -
YC):Y3 = 1: GOTO 2250
2240  IF Y3 > 79 THEN X3 = 143
- (143 - X3) * (79 - YC) / (Y3
- YC):Y3 = 79
2250  IF Y4 < 1 THEN X4 = 143 -
 (143 - X4) * (1 - YC) / (Y4 -
YC):Y4 = 1: GOTO 2270
2260  IF Y4 > 79 THEN X4 = 143
- (143 - X4) * (79 - YC) / (Y4
- YC):Y4 = 79
2270  IF X3 < 11 OR X4 > 275 TH
EN 2380
2280 HF = 1: HCOLOR= 3: HPLOT X
3,Y3 TO X4,Y4
2380 RR = RL:PP = PT:A1 = AS:U1
 = UN:T1 = TN:T2 = TC:X5 = X1:Y
5 = Y1:X6 = X2:Y6 = Y2: RETURN
5080 GZ = VZ:GY = VY + WY:GX =
VX + WX
5090 TC = 5
5100 RT = 3:TP = 5:WR = 50
5500  IF  INT ( RND (1) * 10 +
1) = 1 THEN RL = RL -  SGN (RL)
 + ( INT ( RND (1) * 5 + 1) - 3
)
5505  IF  INT ( RND (1) * 10 +
1) = 1 THEN PT = PT + 3 -  INT
( RND (1) * 2 + 1) * 2
5510  GOSUB 1000: IF PZ < 0 THE
N 5530
5520  GOSUB 2000: GOTO 5500
5530  GOTO 5530
```

A linha 5080 estabelece os valores das variáveis que controlam a posição do aparelho no espaço, de acordo com a sua velocidade e a força e a direção do vento. **GZ** informa sobre a distância percorrida ao longo do eixo **Z** (alto-baixo); **VZ** nos dá a velocidade nesse eixo; **VY**, a velocidade ao longo do eixo **Y** (frente-trás); **WY**, a velocidade do vento nessa mesma direção. Da mesma forma, **GX**, **VX** e **WX** correspondem, respectivamente, à distância percorrida, à velocidade do avião e à velocidade do vento ao longo do eixo **X** (direita-esquerda).

A linha 5090 determina a rotação inicial do motor, a 5100 define os limites da inclinação lateral (**RT**), que faz o avião virar; da inclinação frontal (**TP**), que faz o avião subir ou descer, e da largura da pista (**WR**).

As últimas cinco linhas são comandos temporários. Elas fazem com que o avião voe de maneira irregular, com inclinações laterais e frontais estabelecidas ao acaso.

As sub-rotinas das linhas 1000 a 1120 recalculam todas as variáveis enquanto o aeroplano se movimenta. As linhas 1030 e 1040 verificam se a velocidade caiu abaixo de 30 m por segundo. Quando isso acontece, é chamada a sub-rotina da linha 1500, que recria os efeitos de uma queda ou mergulho. A linha 1510 introduz um elemento de acaso no processo, de forma que tanto podemos mergulhar diretamente para o chão como entrar em parafuso. A sub-rotina 500, que é chamada pela linha 1110, é encarregada de calcular o ângulo em que o aparelho está voando.

A sub-rotina que vai da linha 2000 à 2380 cuida do restante do programa. Ela atualiza a tela, ajustando o painel de instrumentos, que mostra o movimento do avião. Para isso, é necessário apagar o antigo conteúdo de cada instrumento e desenhar os ponteiros correspondentes à nova posição.

O mostrador da velocidade do vento é atualizado pelas linhas 2010 e 2015, enquanto as linhas 2020, 2025, 2030 e 2035 cuidam do mostrador de altitude. O contagiros, por sua vez, é modificado pelas linhas 2040 e 2045; a linha 2050 corrige a direção em que o avião está voando (*bearing*, que é o ângulo da bússola medido em graus). A direção da pista (*runway*, também em graus) é modificada pelas linhas 2060 a 2080, e o novo valor da distância é impresso pela linha 2090.

A linha 2090 também verifica se o horizonte deve ser modificado, e as linhas 2100 a 2170 desenham o novo horizonte artificial no mostrador correspondente. O horizonte verdadeiro é desenhado pelas linhas 2200 a 2280.

Execute então o programa e veja o avião voar sob o controle do piloto automático.

É possível que, após alguns minutos de vôo, a tela gráfica comece a ser preenchida por pontos aleatórios que estragam o desenho. Isso acontece sempre que usamos muitos cordões. No nosso caso, é a variável A$ (responsável pela impressão dos contadores) que é usada a todo instante.

Não se alarme. O próximo artigo resolverá o problema, e permitirá que você assuma o controle do aparelho.



```
500 RA=AD*C:VX=VV*SIN(RA)
510 VY=VV*COS(RA):RETURN
1000 PZ=PZ+GZ:PY=PY+GY:PX=PX+GX
1020 VV=VV+16*(TC*30-VV-8*PT)/V
V
1030 IF ST=1 THEN PMODE 4,1:DRA
W"BM108,40C0":A$="MERGULHO":GOS
UB 4000:DRAW"C5":ST=0:GOTO 1050
1040 IF VV<30 GOSUB 1500
1050 AD=AD+RL:IF AD<0 THEN AD=A
D+360
1060 IF AD>359 THEN AD=AD-360
1070 VZ=VV*SIN(PT*C)-10+VV/15
1080 GZ=VZ:GY=VY+WY:GX=VX+WX
1090 IF VY=0 THEN GD=-PI/2:GOTO
 1110
```

```
1100 GD=-ATN(VX/VY)/C
1110 GOSUB 500
1120 RETURN
1500 PMODE 4,1:ST=1:DRAW"BM108,
40":A$="MERGULHO":GOSUB 4000:PL
AY"T20AGFEDCAGFEDC"
1510 RL=RND(21)-10:PT=-20-RND(5
)
1520 RETURN
2000 PCOPY 5 TO 7:PCOPY 6 TO 8:
PMODE 4,5
2010 LINE(35,120)-(35+20*SIN(VV
*PI/200),120-20*COS(VV*PI/200))
,PSET
2020 TN=PZ/1000:UN=PZ-1000*INT(
TN):LINE(155,120)-(155+15*SIN(T
N*PI/5),120-15*COS(TN*PI/5)),PS
ET
2030 LINE(155,120)-(155+20*SIN(
UN*PI/500),120-20*COS(UN*PI/500
)),PSET
2040 LINE(215,120)-(215+20*SIN(
TC*PI/5),120-20*COS(TC*PI/5)),P
SET
2050 DRAW"BM16,172S8":A$=STR$(A
BS(INT(AD))):GOSUB 4000
2060 IF PY=0 THEN RB=0 ELSE RB=
```



```
ATN(PX/PY)/C:IF PY>0 THEN RB=RB
+180
2070 IF RB<0 THEN RB=RB+360
2080 DRAW"BM80,172":A$=STR$(INT
(RB)):GOSUB 4000:DRAW"BM140,172
":A$=STR$(ABS(INT(PX))):GOSUB 4
000
2090 DRAW"BM196,172":A$=STR$(IN
T(SQR(PY*PY+PX*PX))):GOSUB 4000
2100 YC=120+PT:X1=80:X2=110:Y1=
YC+17*TAN(RL*2*C):Y2=YC-17*TAN(
RL*2*C)
2110 IF (YC<105 OR YC>135)AND R
L=0 THEN 2320
2120 IF Y1<105 THEN X1=95-(95-X
1)*(105-YC)/(Y1-YC):Y1=105:GOTO
 2140
2130 IF Y1>135 THEN X1=95-(95-X
1)*(135-YC)/(Y1-YC):Y1=135
2140 IF Y2<105 THEN X2=95-(95-X
2)*(105-YC)/(Y2-YC):Y2=105:GOTO
 2160
2150 IF Y2>135 THEN X2=95-(95-X
2)*(135-YC)/(Y2-YC):Y2=135
2160 IF X1<80 OR X2>110 THEN 21
80
2170 LINE(X1,Y1)-(X2,Y2),PSET
2180 PMODE 4,1:IF X5-R>10 AND X
5+R<245 AND Y5>0 AND Y5<80 THEN
 CIRCLE (X5,Y5),R,0,10,0,.5
2190 IF RL=RR AND PP=PT THEN 22
90
2200 IF HF=1 THEN LINE (X3,Y3)-
(X4,Y4),PRESET
2210 HF=0:YC=33+PT*4:X3=11:X4=2
44:Y3=YC+118*TAN(RL*2*C):Y4=YC-
118*TAN(RL*2*C)
2220 IF(YC<10 OR YC>79)AND RL=0
 THEN 2290
2230 IF Y3<1 THEN X3=128-(128-X
3)*(1-YC)/(Y3-YC):Y3=1:GOTO 225
0
2240 IF Y3>79 THEN X3=128-(128-
X3)*(79-YC)/(Y3-YC):Y3=79
2250 IF Y4<1 THEN X4=128-(128-X
4)*(1-YC)/(Y4-YC):Y4=1:GOTO 227
0
2260 IF Y4>79 THEN X4=128-(128-
X4)*(79-YC)/(Y4-YC):Y4=79
2270 IF X3<11 OR X4>244 THEN 22
90
2280 HF=1:LINE(X3,Y3)-(X4,Y4),P
SET
2290 WB=AD:IF AD>180 THEN WB=WB
-360
2300 IF RB>180 THEN WB=WB+360-R
B ELSE WB=WB-RB
2310 IF ABS(WB)>60 AND ABS(PY)>
1000 THEN 2370
2320 AN=118/(60*SQR((X3-X4)*(X3
-X4)+(Y3-Y4)*(Y3-Y4))):X5=(X3+X
4)/2+SGN(X3-X4)+WB*AN*(X3-X4)
2330 Y5=(Y3+Y4)/2+2+WB*AN*(Y3-Y
4):IF X5<11 OR X5>244 OR Y5<1 O
R Y5>79 THEN 2370
2340 IF ABS(OY)<1000 THEN R=8-Y
5/10 ELSE R=4000/ABS(PY):IF R*1
0+Y5>80 THEN R=8-Y5/10
2350 IF Y5<10 OR Y5>79 OR X5-R<
11 OR X5+R>244 THEN 2370
2360 CIRCLE (X5,Y5),R,5,10,0,.5
2370 PCOPY 7 TO 3:PCOPY 8 TO 4
2380 RR=RL:PP=PT:RETURN
5080 GZ=VZ:GY=VY+WY:GX=VX+WX
5090 TC=5
5100 RT=3:TP=5:WR=50
5500 IF RND(10)=1 THEN RL=RL-SG
N(RL)+(RND(5)-3)
5505 IF RND(10)=1 THEN PT=PT+3-
RND(2)*2
5510 GOSUB 1000:IF PZ<0 THEN 55
30
5520 GOSUB 2000:GOTO 5500
5530 GOTO 5530
```

A linha 5080 estabelece os valores das variáveis que controlam a posição do aparelho no espaço, conforme a sua velocidade e a força e direção do vento. **GZ** revela a distância percorrida ao longo do eixo **Z** (alto-baixo); **VZ**, a velocidade neste eixo; **VY**, a velocidade ao longo do eixo **Y** (frente-trás); **WY**, a velocidade do vento nessa mesma direção. Por sua vez, **GX**, **VX** e **WX** correspondem, respectivamente, à distância percorrida, à velocidade do aparelho e à velocidade do vento ao longo do eixo **X** (direita-esquerda).

A linha 5090 determina a rotação inicial do motor; a 5100 define os limites da inclinação lateral (**RT**), que faz o avião virar; da inclinação frontal (**TP**), que faz o avião subir ou descer, e da largura da pista (**WR**).

As últimas cinco linhas são comandos temporários. Elas fazem com que o avião voe de maneira irregular, com inclinações laterais e frontais estabelecidas ao acaso.

A sub-rotina das linhas 1000 a 1120 recalcula todas as variáveis enquanto o aeroplano se movimenta. As linhas 1030 e 1040 verificam se a velocidade caiu abaixo de 30 m por segundo. Quando isso acontece, é chamada a sub-rotina da linha 1500, que recria os efeitos de uma queda ou mergulho. A linha 1510 introduz um elemento de acaso no processo, de forma que tanto podemos mergulhar diretamente para o chão como entrar em parafuso. A sub-rotina 500, que é chamada pela linha 1110, é incumbida de calcular o ângulo em que o aparelho está voando.

A sub-rotina que vai da linha 2000 a 2380 cuida do restante do programa. Ela atualiza a tela, ajustando o painel de instrumentos que mostram o movimento do avião.

A linha 2000 transfere a versão original da cabine para outra parte da memória. O mostrador da velocidade do vento é atualizado pela linha 2010, enquanto as linhas 2020 e 2030 cuidam do mostrador de altitude. O contagiros é modificado pela linha 2040, enquanto a 2050 corrige a direção em que o avião está seguindo (*bearing*, que é o ângulo da bússola medido em graus). A direção da pista (*runway*, também em graus) é modificada pelas linhas 2060 a 2080, e o novo valor da distância é impresso pela linha 2090

A linha 2190 verifica se o horizonte deve ser modificado, enquanto as linhas 2100 a 2170 desenham o novo horizonte artificial no mostrador correspondente. O horizonte verdadeiro é desenhado pelas linhas 2200 a 2280.

A nova posição da pista é calculada pelas linhas 2290 a 2360. A linha 2350 verifica se a pista pode ser desenhada na tela (a pista aumenta de tamanho à medida que o avião se aproxima). A linha 2180 apaga a versão antiga quando uma nova é desenhada.

Todas essas modificações foram feitas de maneira invisível, em uma página oculta da tela. A linha 2370 transfere o desenho para a tela usando **PCOPY**, de modo que todas as alterações parecem ocorrer simultaneamente.



Como melhorar a velocidade de execução do simulador de vôo?

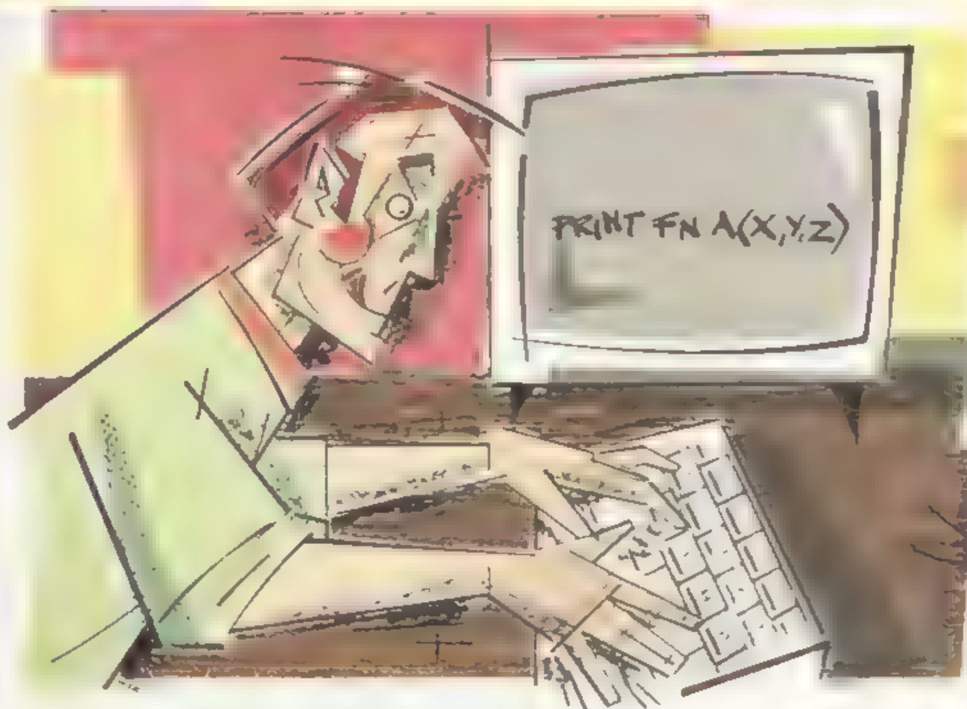
O programa de simulação de vôo é bastante lento. Isso acontece porque ele foi escrito em linguagem BASIC interpretada, com um grande número de equações a serem resolvidas a cada passo da simulação, e de gráficos complexos a serem desenhados. Uma das soluções para o problema seria colocar mais comandos por linha, e tirar espaços em branco; isso, porém, tornaria o programa mais difícil de ser entendido e corrigido, sem aumentar muito a velocidade.

Uma saída (para quem dispõe de um micro com unidade acionadora de disquetes) consiste em utilizar um *compilador* para o programa. Um compilador é um programa especial que traduz um outro programa desenvolvido em linguagem de alto nível (como o BASIC) para código de máquina. Com ele, a velocidade de execução chega a aumentar de dez a trinta vezes, dependendo do modelo usado

Quase todos os micros dispõem de compiladores para a linguagem BASIC. É o caso das linhas Apple, TRS-80 e TRS-Color, por exemplo. O Spectrum e o ZX 81, porém, não contam com esse tipo de recurso. Para a linha MSX, o compilador BASIC é encontrado apenas no exterior.

FUNÇÕES SOB ENCOMENDA

Calcular juros compostos? Encontrar o cubo de um número? Transformar letras maiúsculas em minúsculas? Tudo isso se torna simples quando se trabalha com funções sob encomenda.





Um computador é capaz de resolver problemas matemáticos muito mais rapidamente do que qualquer pessoa. Entretanto, ele só faz o que for programado para fazer, seja por você, seja por um programa pronto, carregado previamente na máquina.

A linguagem mais utilizada para programação de microcomputadores é o BASIC. Em sua maioria, os micros pessoais passam a aceitar comandos nessa linguagem assim que são ligados.

O interpretador BASIC de seu computador conta com uma série de funções matemáticas padronizadas: com elas, o programador pode solicitar cálculos matemáticos complexos, como raiz quadrada, logaritmo, funções trigonométricas e outros. Cada uma dessas funções é definida de tal modo que, assim que reconhece a função, o interpretador toma o valor fornecido para ela pelo programador e executa a operação apropriada. O nome dado às funções evoca as operações que elas realizam: por exemplo, **SQR** (*square root*, ou raiz quadrada, em inglês), **LOG** (logaritmo), **COS** (cosseno) etc.

FUNÇÕES BÁSICAS

Tudo corre bem quando as funções que se pretende usar em um programa existem no BASIC do computador. Muitas funções matemáticas e não matemáticas, porém, não fazem parte do BASIC padrão. Por exemplo, não há nenhuma função chamada **CUBE**, que seria muito útil para calcular automaticamente o cubo de um número (n x n x n).

Mas nem tudo está perdido: felizmente há alternativas para quando o programador precisar de uma função não disponível no BASIC. A primeira seria evitar o emprego de funções no programa: cada vez que um cálculo de cubo precisa ser realizado, por exemplo, pode-se colocar a expressão completa dentro de um comando **LET**. Se isso não for possível — pois o programa ficaria desnecessariamente longo —, pode-se programar uma sub-rotina, que efetuará os cálculos necessários.

A solução mais elegante, entretanto, consiste em um recurso de muitos interpretadores BASIC, que permite a definição de *funções do usuário*. Estas agem como funções de encomenda, que atuam apenas dentro do programa no qual foram definidas. Essa opção existe para os usuários de computadores das linhas Spectrum, TRS-80 (com BASIC nível III), TRS-Color, MSX, Apple II e TK-2000. Os micros das linhas ZX-81 e TRS-80 (baseado em fita cassete) não oferecem essa possibilidade. Em alguns micros, o recurso de definição de funções é bastante poderoso; em outros, como veremos, é mais limitado.

As funções definidas pelo usuário dão ao programador a capacidade de ajustar o BASIC do computador conforme as necessidades de um programa em particular. Os comandos usados para definir uma função diferem ligeiramente de computador para computador, mas seguem em geral esta forma básica:

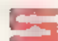
```
DEF FN a (x) = expressão
```

Aqui, a letra **a** é o nome da função (necessário para quando você quiser utilizar essa função em outro ponto do programa, "chamando-a" pela mesma letra). O x entre parênteses é o *parâmetro* da função, ou seja, o valor a ser for-

- FUNÇÕES EM BASIC
- COMO DEFINIR SUAS PRÓPRIAS FUNÇÕES
- QUANTOS PARÂMETROS?
- COMO USAR UMA FUNÇÃO
- FUNÇÕES PARA MANIPULAÇÃO ALFANUMÉRICA
- UMA FUNÇÃO PARA DESENHAR ELIPSES
- FUNÇÕES MAIS RENDOSAS

necido para que a função efetue os cálculos. Alguns computadores (Apple, TK-2000, MSX e TRS-80) permitem dois ou mais parâmetros, enquanto outros (TRS-Color), apenas um.

Para entender o princípio geral de definição e utilização de uma função em um programa, digite e execute o exemplo a seguir. O programa calcula o cubo de um número qualquer:



```
10 CLS
20 PRINT "NUMERO","CUBO"
30 FOR a=1 TO 20
40 PRINT a,FN c(a)
50 NEXT a
60 DEF FN c(x)=x*x*x
```



```
10 DEF FNC(X)=X*X*X
30 PRINT "NUMERO","CUBO"
40 FOR A=1 TO 13
50 PRINT A,FNC(A)
60 NEXT
```

Uma observação interessante é que, no caso do Spectrum, o computador nunca chega a atingir a linha 60, onde a função **FNC** é definida. Dessa forma, a declaração **DEF FN** funciona como uma linha **DATA**. Nos demais computadores, a linha contendo o **DEF FN** deve ser encontrada pelo programa antes que a função correspondente seja chamada. Assim, é conveniente colocá-la logo no início do programa.

Entretanto, os modelos da linha Spectrum procuram automaticamente por linhas contendo **DEF FN**, toda vez que uma função é invocada. Por isso, costuma-se agrupar todos os **DEF FN** no começo do programa, de modo a dar a este maior velocidade de execução.

COMO CHAMAR UMA FUNÇÃO

Para executar uma função, você tem apenas que digitar seu nome (ou seja, a letra usada em sua definição) logo após a expressão **FN**, e no ponto do programa escolhido para utilização do resultado do cálculo elaborado por essa função. Isso pode ser visto na linha 40 do programa para o Spectrum, e na linha 50 do programa para os demais computadores. No exemplo dado, o programa utiliza **FNC(A)**, em vez de **FNC(X)**, como aparece na definição (linha 60 para o Spectrum, e linha 10 para os demais). No caso, **A** (ou **a** para o Spectrum) é o valor do número cujo cubo queremos obter, ou seja, **A** é o *parâmetro* real da função. Como a variável **x**, usada na **DEF FN** serve apenas para especificar como o parâmetro será usado na expressão de cálculo, ela é chamada *pseudoparâmetro*.

Portanto, mesmo que o parâmetro seja diferente a cada chamada da função **FNC** em um programa, a fórmula dada permitirá que se faça o cálculo correto e que o resultado seja retornado ao programa exatamente como as outras funções pré-programadas fazem (**SQR**, **COS**, **LOG** etc.). Assim, a função estabelecida pelo usuário pode ser tratada como se fosse uma espécie de *abreviação* para a fórmula de cálculo especificada em sua definição.

No caso de uma função tão simples como a do exemplo, você poderia pensar que seria mais rápido usar a expres-

são X x X x X do que definir a função só para poder usá-la depois. De fato, este é apenas um exemplo. O uso de funções definidas pelo usuário só será rentável quando elas forem chamadas em vários pontos do programa, de preferência com parâmetros que variem. Aí, a economia é realmente notável. Além disso, **DEF FN** é capaz de trabalhar com expressões matemáticas muito mais longas e complexas do que esta.

Uma boa maneira de aprender como funciona **DEF FN** em um programa é experimentar com o exemplo citado, mudando a definição da função de diversas formas (dividindo o parâmetro por 10, por 2000, calculando a raiz quadrada etc.). Para verificar se a função definida por você calculou corretamente, compare os resultados obtidos por ela com os obtidos pelo uso da mesma fórmula, em modo direto.

O NOME DAS FUNÇÕES

Diversas funções podem ser definidas dentro de um mesmo programa. Cada uma delas deve ter um nome, que varia com o computador. Esse nome é geralmente um caractere (ou cadeia de caracteres) colocado imediatamente após o **DEF FN**. Os micros da linha Spectrum só admitem uma letra. Os demais micros podem usar até dois caracteres. Desse modo, o nome de uma função enfrenta as mesmas restrições que um nome de variável: o primeiro caractere deve ser obrigatoriamente uma letra, e o segundo, uma letra ou um número; além disso, não podem ser utilizados caracteres especiais. Se forem usadas mais de duas letras no nome de uma função, o Spectrum reconhecerá apenas as duas primeiras (no exemplo dado, **DEF FNCUBO** será aceita equivalendo a **DEF FNCU**, e não a **DEF FNC**).

DEFINA OS PARÂMETROS

O próximo passo na definição de uma função consiste em adicionar os parâmetros. Estes são formados pelos valores entre parênteses que devem ser fornecidos para que a função efetue o cálculo. Embora possam ser parecidos com variáveis dentro do **DEF FN**, eles são, de fato, apenas pseudovariáveis, que valem só para a definição.

Tudo o que os parâmetros fazem é "dizer" ao computador qual o número de valores a serem usados no cálculo. Em outras palavras, se você começar uma definição deste modo:

```
DEF FNE (A,B,C) =
```

então o computador esperará que o programa forneça três números toda vez que **FNE** for chamada (um erro de sintaxe ocorrerá na maioria dos micros, se um número errado de parâmetros for fornecido ao se chamar a função). Quando a função for chamada, os três parâmetros do exemplo podem ser oferecidos como constantes ou como variáveis.

Embora os parâmetros usados na **DEF FN** não sejam variáveis, você pode encará-los como tal, ou seja, como uma espécie de variável local, que vale apenas na definição. Eles devem ocorrer na expressão após o sinal de igualdade, com o nome com que foram indicados entre parênteses. Exemplo:

```
DEF FNE (A,B,C)=10 + A * B / C
```

Os parâmetros podem ser variáveis numéricas ou alfanuméricas. Mais adiante, veremos algumas aplicações para funções de tratamento de cadeias alfanuméricas.

Já que não são a mesma coisa, parâmetros e variáveis podem ter os mesmos nomes, sem que isso implique em interferências dos primeiros sobre as segundas e vice-versa.

QUANTOS PARÂMETROS?

A declaração **DEF FN** nos micros das linhas Apple, TK-2000 e TRS-Color é menos versátil do que nos demais computadores, pois permite apenas um parâmetro. Todos os outros valores presentes na expressão aritmética restante precisam ser constantes.

Suponhamos que você queira multiplicar um valor qualquer por 9.81, como parte da função que está definindo. O valor a ser multiplicado poderá receber o nome do único parâmetro disponível, por exemplo, A; a expressão de definição da função será 9.81 x A.

Os outros tipos de computador permitem o uso de quantos parâmetros forem necessários, desde que a definição completa da função esteja contida em uma única linha. Quando a função for chamada, deve-se especificar um número igual de parâmetros, como foi estabelecido na definição, mesmo que haja alguns parâmetros a mais que não foram usados na expressão de definição.

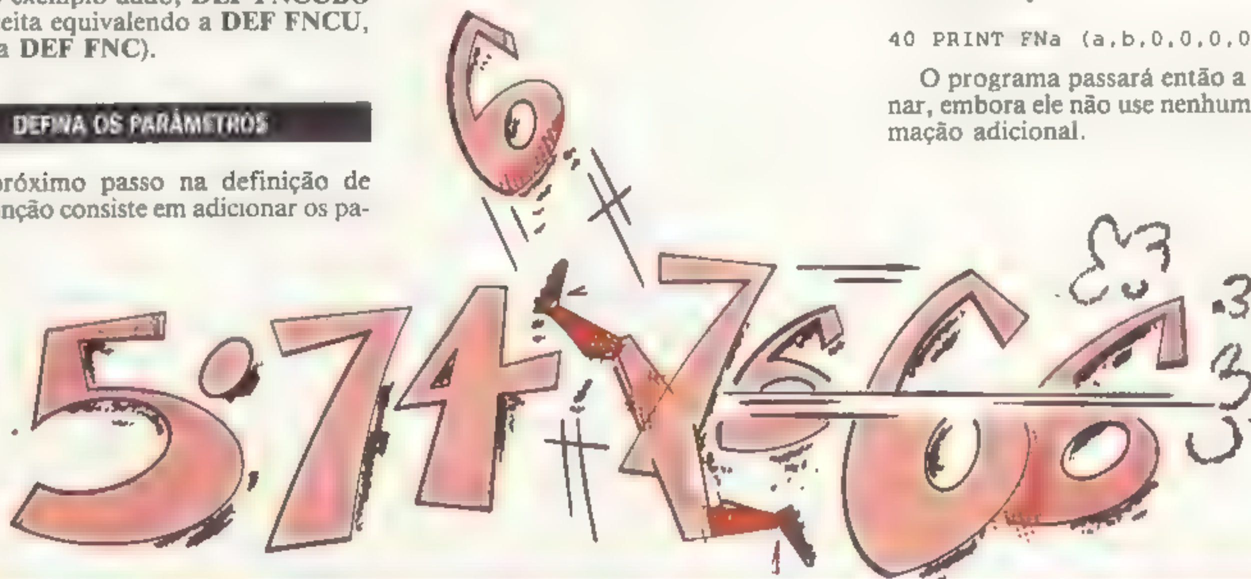
Agora, se você quiser verificar esse fato na prática, digite **NEW** e experimente colocar o programa a seguir (não há uma versão para os micros das linhas TRS-Color, Apple e TK-2000):

```
10 DEF FN a(a,b,c,d,e,f)=a*a*
b
20 PRINT "Introduza dois nume
ros"
30 INPUT a,b
40 PRINT FN a(a,b)
```

Quando você tentar executar o programa acima, verificará que ele acusa um erro na linha 40. Experimente então substituí-la por:

```
40 PRINT FNa (a,b,0,0,0,0)
```

O programa passará então a funcionar, embora ele não use nenhuma informação adicional.



Shift lock

POR QUE USAR UMA FUNÇÃO?

A vantagem das funções em relação às expressões torna-se evidente quando é preciso utilizar uma mesma expressão em vários pontos de um programa. Neste caso, o emprego de funções pré-definidas economiza memória e tempo de digitação e de execução.

A função definida pelo usuário também é útil quando a expressão é muito grande e passível de erros de digitação.

A seguir, apresentamos um exemplo de função matemática que se pode usar repetidas vezes em um programa. Ela arredonda um número qualquer, fornecido por meio de um **INPUT**, fixando o número desejado de casas decimais. Como essa função requer dois parâmetros, deixamos de oferecer uma versão para o Apple, o TK-2000 e o TRS-Color:

```
10 FOR g=0 TO 1 STEP 0
20 PRINT '''"Introduza o nume
ro a arredondar e quantas cas
as decimais voce   deseja"'''
(Pressione <ENTER> a cada vez
)"
30 INPUT numero,casas
40 PRINT '''numero;" com ";ca
sas," casas e' ",FN R(numero,
casas)
50 PAUSE 100: NEXT g
60 DEF FN R(a,b)=INT (a*10^b+
0.5)/10^b
```

```
5 DEF FNR(A,B) =  INT (A * 10^B
+ 0.5)/10^B
10 FOR G=0 TO 1 STEP 0
20 PRINT:PRINT:PRINT
25 PRINT "INTRODUZA O NUMERO A
ARREDONDAR E QUANTAS CASAS
DECIMAIS VOCE DESEJA"
30 INPUT N,CD
40 PRINT:PRINT:PRINT N;" COM ";
CD;" CASAS = ";FNR(N,CD)
50 NEXT G
```

A função **FNR** utiliza uma segunda função em sua definição: é a função **INT**, que obtém a parte inteira de qualquer número. O exemplo mostra também outro aspecto interessante nas funções predefinidas: elas podem abrigar em seu interior outras funções do BASIC, bem como funções definidas pelo usuário. A única limitação, evidentemente, é que não se pode chamar a própria função que está sendo definida.

Embora o TRS-Color, o Apple II e o TK-2000 aceitem apenas funções definidas com um parâmetro, não é difícil fazer bom uso desse recurso mais limitado. Digite e execute o programa a seguir, que define e aplica uma equação para traçar uma elipse. (Ao ser executado, o programa solicitará alguns valores iniciais, como as constantes A e B, e os dois raios da elipse.)

```
10 DEF FNY(X)=SQR(B*R*R-B*X*X/A
)
20 PMODE 3,1
30 PCLS
40 CLS:INPUT"INTRODUZA AS CONST
ANTES A E B ";A,B
50 INPUT "INTRODUZA OS DOIS RAI
OS ";R1,R2
60 R=R1:Y1=FNY(0):R=R2:Y2=FNY(0
)
70 IF Y2>Y1 THEN Y1=Y2
80 IF Y1>95 THEN R1=R1*95/Y1:R2
=R2*95/Y1
90 SCREEN 1,0
100 FOR K=-127 TO 128
110 IF B*R1*R1<B*K*K/A THEN 130
120 R=R1:PSET(K+127,95+FNY(K),2
):PSET(K+127,96-FNY(K),2)
130 IF B*R2*R2<B*K*K/A THEN 150
140 R=R2:PSET(K+127,95+FNY(K),3
):PSET(K+127,96-FNY(K),3)
150 NEXT
160 GOTO 160
```

```
10 DEF FNY(X)=SQR(B*R*R-B*X*X/
   A)
20 HGR
40 HOME:INPUT "INTRODUZA AS CO
   NSTANTES A E B:";A,B
50 INPUT "INTRODUZA OS DOIS RA
   IOS:",R1,R2
60 LET R=R1 : Y1=FNY(0) : R=R2
   : Y2=FNY(0)
70 IF Y2>Y1 THEN Y1=Y2
80 IF Y1>95 THEN R1=R1*95/Y1 :
   R2=R2*95/Y2
100 FOR K= 127 TO 128
110 IF (B*R1*R1)<(B*K*K/A) THEN
    130
115 HCOLOR 3
120 R=R1 : HPLOT K+127,95 + FNY
    (K) : HPLOT K+127,96 - FNY
    (K)
130 IF (B*R2*R2)<(B*K*K/A) THEN
    150
135 HCOLOR 5
140 R=R2 : HPLOT K+127,95 + FNY
    (K) : HPLOT K+127,96 - FNY
    (K)
150 NEXT
```

O programa usa comandos gráficos para desenhar a elipse. A função definida na linha 10 é usada para calcular as coordenadas de cada ponto da elipse. Ela usa internamente uma função matemática predefinida do BASIC: a função de raiz quadrada (**SQR**).

Caso você queira converter esse programa para outros micros, eis aqui a equação das coordenadas da elipse:

$$\frac{X \cdot X}{A} + \frac{Y \cdot Y}{B} = R \cdot R$$

X e Y são as coordenadas horizontal e ver-

tical de cada ponto da elipse, R é o seu raio (na realidade o programa usa dois valores de raio, R1 e R2, para cada elipse), e A e B são constantes que definem o formato da figura.

Alguns microcomputadores, como por exemplo o Spectrum, o MSX e o TRS-80, permitem ainda a utilização de variáveis alfanuméricas em funções definidas pelo usuário.

Se, mesmo depois de tudo o que foi dito, você ainda tem dúvidas a respeito da utilidade de uma função que manipula cadeias alfanuméricas, digite e execute o seguinte programa:

```
10 PRINT "Introduza seu nome"
20 INPUT n$
30 IF CODE n$>90 THEN  LET n$
=FN u$(n$)
40 PRINT ''"Ola,";n$;" - eu n
ao sou esperto ?"
90 DEF FN u$(x$)=CHR$ (CODE x
$ 32)+x$(2 TO )
```

```
5 DEF FNU$(X$) = CHR$ (ASC(X$)
- 32)+MID$(X$,2)
10 INPUT "INTRODUZA  SEU PRIMEI
RO NOME",N$
30 IF ASC(N$)>90 THEN N$ = FNU$
(N$)
40 PRINT:PRINT "OLA, ";N$;", EU
NAO SOU ESPERTO ?"
```

Observe que esse programa permite inicialmente que você entre um nome; em seguida, ele chama uma função definida pelo usuário; essa função imprimirá sucessivamente a primeira letra em maiúscula e as restantes em caracteres minúsculos.

Funções alfanuméricas como essa trabalham da mesma maneira que funções puramente matemáticas. Seus nomes, porém, são diferentes.

Diversamente das funções matemáticas, esses nomes devem conter sempre uma letra seguida de um cifrão, exatamente como acontece com os nomes de variáveis alfanuméricas.

Como se pode perceber pelo exemplo, a função pode ter parâmetros alfanuméricos, que são usados da mesma forma que parâmetros numéricos.

Um outro programa exemplifica melhor este ponto. Trata-se de uma rotina que emprega uma função alfanumérica para abrilhantar as cadeias entradas por meio de comandos **INPUT**:

```
10 FOR f=0 TO 1 STEP 0
20 INPUT "Introduza o titulo
";i$
30 CLS
35 PRINT FN t$(" "+i$)
40 PAUSE 150
50 NEXT f
60 DEF FN t$(x$)=CHR$ 18+CHR$
1+CHR$ 16+CHR$ 2+"***********
*********************"+CHR$
23+CHR$ (16-LEN x$/2)+x$+CHR$
 23+CHR$ 31+"****************
*****************"
```

```
5 DEF FNT$(X$) = STRING$(16-LEN
(X$)/2)+X$+STRING$(16-LEN (X$)
/2)
10 FOR F=0 TO 1 STEP 0
20 INPUT "INTRODUZA O TITULO ";
I$
30 CLS
40 FOR I=1 TO 1000:NEXT I
50 NEXT F
```

Esse programa apresenta algumas particularidades que devem ser especialmente consideradas.

Note, por exemplo, o valor um pouco estranho do comando **STEP** na linha 10. Na verdade, o que ele faz é causar um laço de repetição infinita, informando diretamente ao computador que o incremento do comando **FOR** é 0; isso quer dizer que ele nunca irá alcançar o valor final 1.

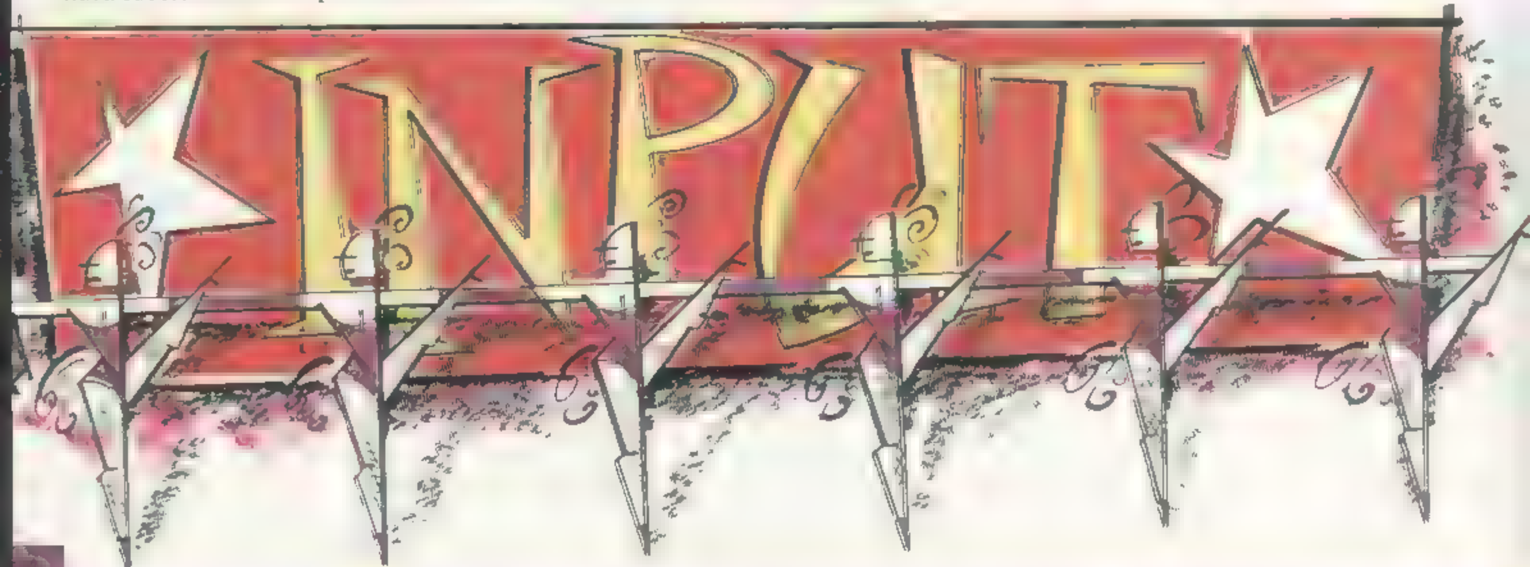
Embora neste caso seja mais fácil recorrer a um comando **GOTO**, é conveniente que você saiba que programas melhor estruturados evitam o **GOTO** para trás, o que seria considerado má prática de programação.

FUNÇÕES RENDOSAS

Eis aqui outro exemplo do comando **DEF FN** em ação. Trata-se de um programa relativamente simples, que pode ser empregado em tarefas como calcular os juros de sua caderneta de poupança.

Ele define uma função que calcula rigorosamente o valor a ser atingido por um investimento após um dado período de tempo, e levando em consideração os juros pagos por ano.

```
10 DEF FN c(t)=INT (AM*((R/
100+1)^T)*100)/100
20 INPUT "Quantos cruzados? "
;AM
30 INPUT "Taxa de juros (%)?
";R
40 INPUT "No  de unidades de
tempo? ",T
50 PRINT AT 9,0,"Quantia tota
```



```
1 apos a operacao= ",TAB 12,
FN C(T)
60 PRINT  INVERSE 1,AT 20,1,"
Qualquer tecla para recomecar
"
70 PAUSE 0  CLS   GOTO 20
```



```
10 DEF FN C(T)=INT(AM*((1+R/100
)^T)*100)/100
20 INPUT"QUANTOS CRUZADOS ";AM
30 INPUT"TAXA DE JUROS (%) ",R
40 INPUT"QUANTAS UNIDADES DE TE
MPO ",T
50 PRINT"QUANTIA TOTAL APOS APL
ICACAO =",FNC(T)
60 PRINT PRINT"QUALQUER TECLA P
ARA RECOMECAR"
70 A$=INKEY$ IF A$="" THEN 70
80 GOTO 20
```

Para o programa funcionar no Apple e no TK-2000, substitua a linha 70 por:

```
70 GET A$
```

O programa funciona de forma muito simples. Definida a função, os valores de entrada são colocados por meio de uma série de comandos **INPUT**, de modo que o usuário possa fornecer ao computador os detalhes da poupança.

As variáveis que o programa usa são: **T**, para o período de tempo em questão; **AM**, para a quantia inicial em dinheiro a ser aplicada; e **R**, para a taxa de juros, em % ao ano.

O único parâmetro da função **FNC** é **T**. Isso pode ser percebido facilmente a partir da própria definição da função, na linha 10. Entretanto, a expressão matemática na mesma linha contém três variáveis, que são **AM**, **R** e **T** — e apenas uma delas é o parâmetro. Esta é uma característica muito útil das funções definidas pelo usuário. Ela permite que se misturem parâmetros e variáveis na mesma expressão, evitando que seja necessário definir todos os elementos como parâmetros.

Esta é uma maneira de se contornar a limitação de um parâmetro por função nos microcomutadores das linhas TRS-Color, Apple e TK-2000.

Já no MSX é possível até mesmo definir uma função inteiramente sem parâmetros (neste caso, não coloque parênteses, nem em **DEF FN**, nem na chamada a função)

Os usuários do Spectrum podem estranhar o fato de que a expressão na linha 10 usa variáveis ainda não definidas, o que deveria provocar uma mensagem de erro. Entretanto, como foi dito linhas atrás, no Spectrum o processo funciona de modo diferente; neste caso, a linha 10 é procurada *depois* que as variáveis foram definidas pelos comandos **INPUT** do programa.

Os juros calculados aqui são juros compostos — isso quer dizer que o valor ganho em um período é usado para calcular os juros do período seguinte (juros sobre juros).

OPERAÇÕES ARITMÉTICAS

Isso é levado em consideração automaticamente pela fórmula da função, simplificando o cálculo dos números procurados. Veja a seguir como se processa esse cálculo na prática.

Se, por exemplo, a taxa de juros for 10% após um período de tempo, o programa calculará o valor original mais 0.1 multiplicado por esse valor. Você pode ver isto na expressão contida na função, como R/100 + 1. Assim, no exemplo dos 10%, o resultado após um período de aplicação é de 1.1 vezes o valor original aplicado. Após dois períodos, será 1.1 vezes o novo valor, e assim por diante. Em **T** períodos, isso dará 1.1 elevado a **T**. Esta é a segunda parte da função, que toma a forma completa vista no programa, ou seja, **(R/100 + 1)T**. A parte restante da expressão multiplica isto pelo valor do principal (**AM**) e arredonda o resultado final para duas casas decimais, multiplicando por 100, tomando o seu **INT**eiro, e dividindo por 100.

Tudo se resume, portanto, a simples operações aritméticas.

Você poderá também definir funções trigonométricas que não existem na maioria dos interpretadores BASIC, como arco seno e arco co-seno.

```
ASN(X)=ATN(X/SQR( X*X+1))
ACS(X)=-ATN(X/SQR(-X*
X+1))+1 5708
```

Agora, procure treinar, "inventando" algumas funções novas para outras áreas da matemática.

UM EDITOR DE TEXTOS (3)

A procura da palavra certa foi sempre um elemento estimulante para os grandes escritores. Esta lição ensina como fazer isso com a prosaica ajuda de um computador.

As duas primeiras partes desta lição apresentaram as funções básicas de edição na tela, que permitem criar textos e armazená-los. A terceira e última parte fornece as rotinas de ordenação, procura de palavras, impressão e uso de "máscara de texto".

ORDENAÇÃO

Baseada na rotina de troca retardada, a rotina de ordenação é usada para colocar linhas em ordem alfabética. Ela é de grande utilidade para ordenar nomes ou outros dados.

PROCURA DE PALAVRAS

Essa rotina "varre" o texto em busca de uma determinada palavra ou fragmento de palavra. Ela deve ser ativada dentro do modo editor. A procura começa no ponto onde o marcador está localizado. Por isso, certifique-se de que ele está na posição mais adequada para dar início à busca.

Há dois motivos básicos para que não se ache a palavra procurada: ou ela foi digitada de modo errado ou está dividida em duas linhas.

Quando a palavra for encontrada, o programa continuará no modo editor e a linha poderá ser copiada imediatamente na área de trabalho.

IMPRESSÃO

A rotina de impressão possibilita o envio do texto gerado para a impressora. Ela apresenta algumas características especiais, incluindo uma rotina para controlar o formato das cartas e outra para uso do texto com "máscara", permitindo que você adicione partes variáveis ao texto.

Se a sua impressora trabalha com comandos não padronizados, pode ser necessário algum ajuste no programa. Preste particular atenção aos caracteres que provocam avanço de linha (*linefeed*, ou **LF**) e retorno do carro (*carriage return*, ou **CR**). O programa usa o caractere ASCII 13 para ambas as funções.

FORMATAÇÃO

Pouco adiantaria poder escrever e corrigir um texto se não fosse possível imprimi-lo na forma desejada. Pode ser necessário, por exemplo, colocar o cabeçalho da página no centro de uma linha, seguida por outras linhas em branco. Isso se torna fácil quando se usam os comandos de formatação. Outro exemplo comum é quando se quer colocar a data ou o endereço do remetente na margem direita e o endereço do destinatário à esquerda.

Os símbolos empregados na rotina de formatação são os mesmos do programa do artigo *Escreva Cartas sem Esforço* (página 17). Lembre-se de que eles devem aparecer sempre no início da linha. Esses símbolos são os seguintes: o sustenido ("#"), que coloca a linha de texto na margem direita da página (se existir só uma linha com essa marca, ela será colocada com o último caractere na última posição da linha; se houver mais linhas, o programa procurará a mais longa e alinhará as outras com base nesta); o ampersand ("&"), que faz o texto que o segue ser impresso no início da linha seguinte (por exemplo, ele pode ser usado em cartas, para posicionar cada linha do endereço do destinatário); o cifrão ("$"), que, além de passar o texto para outra linha, provoca um recuo na primeira palavra da nova linha; o asterisco ("*"), que coloca o texto no centro da linha. Ao usar essa rotina, tenha cuidado com linhas de texto muito longas. Estas devem ter, no máximo, a mesma largura da linha de impressão.

COMO USAR "MÁSCARA DE TEXTO"

Além dos comandos de formatação, o programa dispõe de outro recurso (inexistente no do Spectrum) com o qual se pode criar uma carta padrão e deixar que algumas partes específicas sejam variáveis. Esse recurso é o "símbolo de máscara". Ele opera do seguinte modo: quando você quiser inserir uma frase no texto, coloque um par de colchetes ("][") na posição escolhida. No momen-



to da impressão, esse sinal será substituído pelo bloco de texto apropriado. Dessa maneira, você pode começar a carta com um "Caro][" e só digitar o nome do destinatário quando lhe convier.

O "símbolo de máscara" pode ser colocado em qualquer posição no texto, menos depois de um sustenido (#). É que o programa mede primeiro o tamanho de cada linha que vai ser posicionada à direita, determinando a seguir onde deve começar a impressão dessas linhas. Como essa operação é feita antes de se carregar os blocos variáveis, sempre surge algum problema.

O texto que substitui os colchetes é digitado diretamente do teclado na hora da impressão, ou lido de um arquivo criado pelo próprio programa.

O máximo de caracteres admitido por linha no TRS-Color é 32. No MSX, esse número sobe para 39 e, no Apple, para 40. O texto será quebrado, assim, em unidades de 32 a 40 caracteres, dependendo da máquina. Cada par de colchetes permite a entrada de, no máximo, 250 caracteres.

Se os blocos variáveis de texto forem carregados de um arquivo, o computador se encarregará de colocar os dados nos seus devidos lugares. Cuide apenas para que os dados estejam na ordem correta e para que haja informação para todos os pares de colchetes.

S

```
4000 REM imprimir
4010 LET tt=(pl 11)/2
4020 LET d=0
4025 FOR n=t+1 TO b 1
4030 LET a$=t$(n)
4032 IF LEN a$=0 THEN  NEXT n
RETURN
4034 IF a$(LEN a$ 1)<>CHR$ 32 T
HEN  GOTO 4037
4035 IF a$(LEN a$) CHR$ 32 THEN
 LET a$=a$( TO LEN a$ 1)  GOTO
 4032
4037 LET l LEN a$
4040 LET c=0
4050 IF c l THEN  NEXT n  LPRIN
T CHR$ 13: RETURN
4060 LET c=c+1  LET d=d+1  IF c
.. THEN  GOTO 4.00
4070 IF a$(c)="#" THEN  GOTO 45
00
4080 IF a$(c)="*" THEN  GOTO 47
00
4085 IF a$(c) "&" THEN  GOTO 48
00
4 [illegible] IF a$(c)="$" THEN  LPRINT
CHR$ 13,CHR$ 13,  LET d=0  GOTO
 4 [illegible]00
4 [illegible] LET n n+.  IF n>=b  THEN
 LET l= EN a$  GOTO 41..
41 5 IF t$(n,1)="$"  R t$ n,1 =
"#" OR t$(n, )="*" OR t$ [illegible] ,)="
&" THEN  GOTO 4110
4 [illegible] LET a$ a$+t$(n
41 [illegible] IF a$ LEN a$ 1,<>CHR$ 32 T
HEN  [illegible] 4100
4 [illegible] IF a$(LEN a$ =CHR$ 32 THEN
 LET a$=a$( TO LEN a$ 1)
4 [illegible] GOTO 4 07
4 [illegible] LET n n 1  LET l LEN a$
4 [illegible] IF a$( )=CHR$ 32 THEN  GOT
O 48 0
[illegible] LPRINT a$ c,.
[illegible] IF d>  THEN  LET d 0
4 [illegible] GOTO 4050
4500 LET nl=0: LET ta=11: LET b
e )
4510 LET le=LEN a$ 1  IF le>11
THEN  PRINT  FLASH  ,"ERRO NO F
ORMATO  ENDERE O MUITO LONGO"
SOUND 2,.0  RETURN
4520 IF le>be THEN  LET be=le
4530 LET nl=nl+1: LET n=n+1: LE
T a$ t$(n)
4532 IF LEN a$=0 THEN  NEXT n:
RETURN
4535 IF a$(LEN a$)=CHR$ 32 THEN
 LET a$=a$( TO LEN a$-1): GOTO
 4532
4538 IF a$(1, "#" THEN  GOTO 45
10
4540 LET n=1
4550 LET tr=tt+11-be  FOR g=1 T
O nl  FOR h=1 TO tr  LPRINT CHR
$ 32,  NEXT h  LET n=n+1  LET a
$=t$(n)
4552 IF LEN a$=0 THEN  NEXT n:
RETURN
4555 IF a$(LEN a$)=CHR$ 32 THEN
 LET a$=a$( TO LEN a$ .)  GOTO
 4552
4558 LPRINT a$(2 TO ,  NEXT g
4560 NEXT n  RETURN
4700 LET ta=(11 l.)/2+tt  IF ta<
tt THEN  LPRINT CHR$ 13  PRINT
 FLASH 1;"ERRO NO FORMATO - IMP
OSSIVEL  CENTRALIZAR"  SOUND
2,10  RETURN
4710 LPRINT CHR$ 13;: FOR m=1 T
O ta  LPRINT CHR$ 32,  NEXT m
LPRINT a$(2 TO ):  LET d=0  NEX
T n: RETURN
4800 LET sl=11 d-1  LET cc=c+1
```




```
 LET x=1
4810 IF cc>=1 THEN  GOTO 4825
4820 IF a$(cc)<>CHR$ 32 THEN  L
ET cc=cc+1: LET x=x+1: GOTO 481
0
4825 IF x>=11 THEN  LPRINT CHR$
 13: PRINT  FLASH 1;"ERRO NO FO
RMATO - PALAVRA MUITO GRANDE":
SOUND 2,10: RETURN
4840 IF s>=x THEN  GOTO 4112
4850 LPRINT CHR$ 13;: LET d=0
4900 FOR m=1 TO tt: LPRINT CHR$
 32;: NEXT m: GOTO 4050
8000 REM procura
8002 IF z$="" THEN  PRINT #1;AT
 0,0; BRIGHT 1;"Nao foi definid
a a palavra alvo": PAUSE 100: P
RINT #1;AT 0,0;s$;s$: RETURN
8005 PRINT #1;AT 0,0;s$;s$: IF
p=b-2 THEN  LET p=4
8010 FOR n=1 TO 33-LEN z$
8020 IF t$(p,n TO n+LEN z$-1)=z
$ THEN  LET n=33-LEN z$: NEXT n
: GOTO 8050
8030 NEXT n
8040 LET p=p+1: IF p=b-2 THEN
LET p=p-1: GOTO 8050
8045 GOTO 8010
8050 LET p=p+1: GOSUB 1000: RET
URN
8500 REM ordenacao
8505 PRINT #1;AT 0,0;s$;s$
8510 LET ss=4
8520 IF t$(ss,1)="^" THEN  GOTO
 8550
8530 LET ss=ss+1: IF ss=b THEN
 PRINT #1;AT 0,0; BRIGHT 1;"lim
ites nao definidos": PAUSE 100:
 PRINT #1;AT 0,0;s$;s$: RETURN
8540 GOTO 8520
8550 LET se=ss+1
8560 IF t$(se,1)="^" THEN  GOTO
 8600
8570 LET se=se+1: IF se=b THEN
 PRINT #1;AT 0,0; BRIGHT 1;"Som
ente um limite foi definido": P
AUSE 100: PRINT #1;AT 0,0;s$;s$
: RETURN
8580 GOTO 8560
8600 IF ss=se-1 OR ss=se-2 THEN
  GOTO 8900
8610 PRINT #1;AT 0,0; BRIGHT 1;
"Ordenando"
8620 FOR i=ss+1 TO se-1
8630 LET k=i
8640 FOR j=i+1 TO se-1
8650 IF t$(j)<t$(k) THEN  LET k
=j
8660 NEXT j: IF i<>k THEN  LET
w$=t$(k): LET t$(k)=t$(i): LET
t$(i)=w$
8670 NEXT i
8900 FOR n=ss TO b: LET t$(n)=t
$(n+1): NEXT n
8910 FOR n=se-1 TO b: LET t$(n)
=t$(n+1): NEXT n: LET b=b-2: IF
 p>b-2 THEN  LET p=p-2
8915 PRINT #1;AT 0,0;s$;s$
8930 GOSUB 1000
8940 RETURN
```

Para usar a rotina de ordenação, marque os pontos correspondentes ao início e ao fim da operação, colocando uma flecha (↑) acima da primeira linha a ser ordenada e outra abaixo da última. Para iniciar o processo, tecle **[CAPS SHIFT]** e 4. As flechas serão eliminadas durante a ordenação.

Para fazer a operação de busca, tecle **[CAPS SHIFT]** e 2; a palavra a ser procurada deve ser digitada conforme requisitado. A busca começará imediatamente. Assim que o computador encontrar a primeira ocorrência, o cursor aparecerá sob ela. Se quiser continuar a procura mais adiante, pressione **[CAPS SHIFT]** e 3. Para alterar a palavra, tecle **[CAPS SHIFT]** e 2.

Tecle 6 para entrar na rotina de impressão da linha 4000. Isso fará o texto ser impresso de acordo com o que foi definido na rotina da linha 100, ou seja, 32 caracteres por linha e 32 linhas por página. Se quiser alterar esses valores, tecle 7 e não 6.

Na formatação do texto, coloque o sinal adequado, para que a linha seja impressa do modo desejado. O sinal "#" indica que a linha deve ser posicionada junto à margem direita do texto. O "&" provoca uma mudança de linha e inicia a impressão na margem esquerda. O "$" faz com que haja um avanço de duas linhas e não de uma apenas. O "*" centraliza o texto.



```
3000 CLS:PRINT@7,BL$;"IMPRESSAO
";BL$
3010 IF TL<2 THEN 3050
3020 PRINT"DA (M)EMORIA OU DE (
A)RQUIVO ?"
3030 R$=INKEY$:IF R$<>"M" AND R
$<>"A" THEN 3030
3040 IF R$="M" THEN 3060
3050 GOSUB 4500
3060 IF TL=1 THEN PRINT "(nao ha
 arquivo":PYS "T2003EDCA":GOTO
3570
3070 KF=0:PRINT "QUER USAR MASC
ARA (S/N) ?"
3080 R$=INKEY$:IF R$<>"S" AND R
$<>"N" THEN 3080
3090 IF R$="N" THEN 3150
3100 PRINT:PRINT"CARREGA BLOCOS
 VARIAVEIS          (T)ECLADO OU
```

```
 A)RQUIVO?"
3110 R$ INKEY$ IF R$<>"T" AND R
$ "A" THEN 3110
3120 KF=2 IF R$="T" THEN KF=1 G
OTO 3150
3130 PRINT:LINEINPUT "INTRODUZA
 O NOME DO ARQUIVO ",VB$
3140 IF LEFT$ VB$,1)<"A" OR LEF
T$(VB$,1)>"Z" THEN 3130
3150 CLS PRINT "DESEJA REPROGRA
MAR A IMPRESSORA (S/N) ?"
3160 R$=INKEY$ IF R$<>"S" AND R
$<>"N" THEN 3160
3170 IF R$="S" GOSUB 5500
3180 CLS
3190 VB=0 PP=0 AZ=0 LC=1 PRINT
"DESEJA IMPRIMIR NA TELA (S/N)?
" PRINT"enter RETORNA AO MENU P
RINCIPAL"
3200 R$=INKEY$:IF R$<>"S" AND R
$<>"N" AND R$<>CHR$(13) THEN 32
00
3210 IF R$=CHR$(13) THEN RETURN
3220 IF KF=0 THEN 3240
3230 IF DL=1 AND KF=2 THEN FREA
D VB$,FROM 0,DV  FREAD VB$,DV E
LSE IF K=2 THEN OPEN "I",#-1,VB
$:INPUT#-1,DV,DV
3240 P=0 GP$="" IF R$="N" THEN
P=-2 GP$=STRING$ GP,32)
3250 FOR K=1 TO TL 1·IF LEFT$(T
X$(K),1)="#" AND LEN(TX$(K))-1>
AZ THEN AZ=LEN(TX$(K))
3260 NEXT IF AZ>TW THEN PRINT"e
rro endereco muito longo"·PY$="
T402AB"·GOTO 3570
3270 K=1 PRINT#P,LF$,GP$;·A$=""
 IF AZ>0 THEN AZ$=STRING$(GP+TW
-AZ,32)
3280 TT$=TX$(K)
3290 IF TT$="" THEN PRINT #P,CH
R$(13),GP$, PP=0,LC=LC+1·GOSUB
3590 GOTO 3520
3300 BP=INSTR(TT$,"][")·IF BP=0
 OR KF=0 THEN 3390
3310 IF KF=1 THEN 3370
3320 IF DL=1 THEN 3360
3330 IF EOF(-1) THEN 3350
3340 INPUT#-1,RP$·GOTO 3380
3350 PRINT"erro falta de dados
no arquivo":PY$="L2005DL402D":G
OTO 3570
3360 IF EOF(VB$) THEN 3350 ELSE
 FIREAD VB$,RP$ GOTO 3380
3370 BL=BL+1 PRINT.PRINT "INTRO
DUZA BLOCO VARIAVEL",BL,"?";:LI
NEINPUT RP$
3380 TT$=LEFT$(TT$,BP-1)+RP$+MI
D$ TT$,BP+2) GOTO 3300
3390 ON INSTR ("&$*#",LEFT$(TT$
,1)) GOTO 3460,3470,3490,3510
3400 IF PP+LEN(TT$)<=TW THEN PR
INT#P,TT$, PP=PP+LEN(TT$),GOTO
3520
3410 TA$=LEFT$(TT$,TW-PP)
3420 IF INSTR(TT$," ")>TW THEN
PRINT "erro·palavra muito grand
e na",TT$·PY$="T1002CB"·GOTO 35
70
3430 IF RIGHT$(TA$,1)=" " THEN
3450
3440 IF LEN(TA$)>0 THEN TA$=LEF
T$(TA$,LEN(TA$)-1)·GOTO 3430
3450 PRINT#P,TA$,CHR$(13);GP$,:
PP=0:LC=LC+1:GOSUB 3590:TT$=MID
$(TT$,LEN(TA$)+1):IF TT$<>"" TH
EN BP=1·GOTO 3400 ELSE 3520
3460 PRINT#P,CHR$(13),GP$: PP=0
·LC=LC+1·GOSUB 3590·TT$=MID$(TT
$,2):GOTO 3300
3470 TT$=MID$(TT$,2):PRINT#P,CH
R$(13);GP$;STRING$(INT(TW/4),32
),·PP=INT(TW/4)
3480 LC=LC+1:GOSUB 3590:GOTO 33
00
3490 TT$=MID$(TT$,2):IF LEN(TT$
)>TW THEN PRINT"erro·impossivel
 centralizar";TT$ PY$="T103C":G
OTO 3520
3500 PRINT#P,CHR$(13);GP$,STRIN
G$(INT((TW·LEN(TT$))/2),32);TT$
,CHR$(13),GP$;·PP=0:LC=LC+1:GOS
UB 3590·GOTO 3520
3510 PRINT#P,CHR$(13);AZ$;MID$(
TT$,2);·PP=0·LC=LC+1·GOSUB 3590
3520 K=K+1·IF P=0 THEN FOR Z=1
TO 500·NEXT
3530 IF K<TL THEN 3280
3540 IF P=-2 THEN PRINT #P,LF$;
LF$ ELSE PRINT:PRINT
3550 IF K=1 THEN CLOSE #-1 ELSE
 IF KF=2 THEN CLOSE
3560 IF P=0 THEN 3190 ELSE RETU
RN
3570 FOR Z=1 TO 10:PLAY PY$:NEX
T:IF KF=1 THEN CLOSE#-1 ELS IF
KF=2 THEN CLOSE
3580 RETURN
3590 IF LC>TH THEN PRINT #P,LF$
,LF$;GP$; LC=1
3600 RETURN
5070 L=CP·PRINT @384,"INTRODUZA
 PALAVRA PROCURADA"
5080 LINEINPUT TG$·IF TG$="" TH
EN 5070
5090 PRINT @500,"procurando";BL
$;
5100 IF L=TL THEN CP=TL:CLS:GOS
UB 2090 RETURN
5110 IF INSTR(TX$(L),TG$)=0 THE
N L=L+1:GOTO 5100
5120 CP=L+1:CLS:GOSUB 2090:RETU
RN
5130 IF SS>SE THEN TT=SS: SS=SE
: SE=TT
5140 SE=SE-1
5150 PRINT@500,"ordenando";BL$:
5160 FOR I=SS TO SE-1
5170 K=I
5180 FOR J=I+1 TO SE
5190 IF TX$(J)<TX$(K) THEN K=J
5200 NEXT·IF I<>K THEN TT$=TX$(
K)·TX$(K)=TX$(I)·TX$(I)=TT$
5210 NEXT:CLS:GOSUB 2090:RETURN
```

Para fazer funcionar a rotina de ordenação, entre no modo editor e leve o marcador ">" para um dos extremos do bloco a ser ordenado. Então, tecle "@". Em seguida, mova o marcador, levando-o para o outro extremo, e tecle "@" novamente. Essa operação inicia automaticamente a ordenação.

A função de procura de palavras é ativa, dentro do modo editor, quando se pressiona a tecla P. Você responde qual a palavra a ser procurada por intermédio da mensagem mostrada na área de trabalho. Pressione a tecla [**RETURN**] para dar início à busca. Quando a palavra desejada for encontrada, o marcador será colocado logo abaixo da linha que a contém.

Para imprimir um texto já editado, tecle I a partir do menu principal. Você terá então de responder a várias perguntas. A primeira delas indaga se o texto a ser impresso está na memória ou no arquivo. Caso você escolha imprimir o que está na memória e não haja nada aí, um sinal sonoro será ativado e uma mensagem mostrada na tela do computador; o programa voltará então ao menu principal.

Em seguida, será oferecida a opção de mudar os parâmetros da impressora. Responda com S ou N. Essa rotina torna possível a alteração dos valores pré-determinados, que são os seguintes: 80 para a largura do formulário, 60 para a largura da linha de texto (deixando margens de dez colunas), 66 para o número de linhas total por formulário e 60 para o número de linhas de texto (deixando três linhas de espaço no topo e no fim da página).

Assumidos quando o programa é executado, os valores indicados acima serão mantidos até que você os altere.

A última pergunta diz respeito a uma cópia do texto na tela. Se você responder S, o texto será imediatamente colocado na tela; um N se encarrega de enviá-lo para a impressora.

Os sinais empregados para formatação do texto são os mesmos que aparecem no programa do já citado artigo da página 17. O sustenido ("#") faz com que a linha seja levada para a margem direita, enquanto o cifrão ("$") provoca um avanço de uma linha e um recuo do texto subseqüente.

O ampersand ("&") determina um avanço de linha, fazendo com que o texto a seguir seja impresso no início da linha abaixo (sem recuo). O asterisco ("*") centraliza a linha de texto.

Para inserir blocos de texto variáveis dentro do texto principal, utilize um par de colchetes ("][") na posição desejada. Responda S quando perguntado se deseja recorrer à "máscara de texto". Os blocos podem ser lidos a partir do teclado ou de um arquivo. Se sua opção for pelo teclado você deve digitar cada bloco quando o programa pedir. No caso de escolher o arquivo, essa operação será feita automaticamente — o computador lerá o texto de um arquivo previamente preparado para isso (use o próprio programa).



```
3000 CLS:BL$="Imprimir":GOSUB 220
3010 IF TL<2 THEN 3050
3020 PRINT:PRINT:PRINT"Imprime texto da [M]emória ou [F]ita?";
3030 R$=INKEY$:IF R$<>"M" AND R$<>"F" THEN 3030
3040 IF R$="M" THEN 3060
3050 GOSUB 4500:CLS:BL$="Imprimir":GOSUB 220
3060 IF TL=1 THEN BEEP:LOCATE 8,12:PRINT"Não há texto na memória.":FOR I=1 TO 1000:NEXT:RETURN
3070 KF=0:PRINT:PRINT"Quer usar máscara? (S/N)";
3080 R$=INKEY$:IF R$<>"S" AND R$<>"N" THEN 3080
3090 IF R$="N" THEN 3150
3100 PRINT:PRINT"Carrega os blocos variáveis do [T]eclado ou da [F]ita?";
3110 R$=INKEY$:IF R$<>"T" AND R$<>"F" THEN 3110
3120 KF=2:IF R$="T" THEN KF=1:GOTO 3150
3130 PRINT:LINEINPUT"Nome do arquivo ";VB$
3140 IF LEFT$(VB$,1)<"A" OR LEFT$(VB$,1)>"Z" THEN 3130
3150 PRINT:PRINT"Altera a configuração da impressora? (S/N)";
3160 R$=INKEY$:IF R$<>"S" AND R$<>"N" THEN 3160
3170 IF R$="S" THEN GOSUB 5500
3180 CLS:BL$="Imprimir":GOSUB 220
3190 VB=0:PP=0:AS=0:LC=1:PRINT:PRINT:PRINT"Quer uma demonstração na tela? (S/N)":PRINT"Tecle <RETURN> para retornar ao menu";
3200 R$=INKEY$:IF R$<>"S" AND R$<>"N" AND R$<>CHR$(13) THEN 3200
3210 IF R$=CHR$(13) THEN RETURN
3220 IF KF<>2 THEN 3240
3230 OPEN VB$ FOR INPUT AS #1:INPUT#1,DV,DV
3240 P=0:GP$="":IF R$="N" THEN P=-1:GP$=STRING$(GP,32)
3250 FOR K=1 TO TL-1:IF LEFT$(TX$(K),1)="#" AND LEN(TX$(K))-1>AS THEN AS=LEN(TX$(K))
3260 NEXT:IF AS>TW THEN PRINT:PRINTTAB(10);"Endereço muito longo!":GOTO 3610
3270 K=1:A$="":IFAS>0 THEN A$=STRING$(GP+TW-AS,32)
3280 IF NOT P THEN PRINT:PRINT:PRINT:ELSE LPRINT LF$;GP$;
3290 TT$=TX$(K)
3300 IF TT$<>"" THEN 3320 ELSE IF P THEN LPRINT L1$;GP$ ELSE PRINT
3310 PP=0:LC=LC+1:GOSUB 3630:GOTO 3560
3320 BP=INSTR(TT$,"]["):IF BP=0 OR KF=0 THEN 3390
3330 IF KF=1 THEN 3370
3340 IF EOF(1) THEN 3360
3350 INPUT#1,RP$:GOTO 3380
3360 PRINT:PRINTTAB(3)"Fim de dados para blocos variáveis ":GOTO 3610
3370 BL=BL+1:PRINT:PRINT"Bloco variável",BL;"?";:LINEINPUT RP$
3380 TT$=LEFT$(TT$,BP-1)+RP$+MID$(TT$,BP+2):GOTO 3320
3390 ON INSTR("&$*#",LEFT$(TT$,1)) GOTO 3470,3490,3510,3540
3400 IF PP+LEN(TT$)<=TW THEN IF P THEN LPRINT TT$;CHR$(32);:PP=PP+LEN(TT$):GOTO 3560 ELSE PRINTTT$;CHR$(32);:PP=PP+LEN(TT$):GOTO 3560
3410 TA$=LEFT$(TT$,TW-PP)
3420 IF INSTR(TT$," ")>TW THEN PRINT:PRINT"Palavra muito longa em:":PRINTTT$:GOTO 3610
3430 IF RIGHT$(TA$,1)=CHR$(32) THEN 3450
3440 IF LEN(TA$)>0 THEN TA$=LEFT$(TA$,LEN(TA$)-1):GOTO 3430
3450 IF P THEN LPRINT TA$;L1$;GP$; ELSE PRINTTA$
3460 PP=0:LC=LC+1:GOSUB 3630:TT$=MID$(TT$,LEN(TA$)+1):IF TT$<>"" THEN BP=1:GOTO 3400 ELSE 3560
3470 IF P THEN LPRINT L1$;GP$; ELSE PRINT
```

```
3480 PP=0:LC=LC+1:GOSUB 3630:TT
$=MID$(TT$,2):GOTO 3320
3490 TT$=MID$(TT$,2): IF P THEN
LPRINT L1$,GP$,STRING$(TW/4,32)
: ELSE PRINT LOCATE TW/4
3500 PP=INT(TW/4):LC=LC+1:GOSUB
 3630:GOTO 3320
3510 TT$=MID$(TT$,2): IF LEN(TT$
)>TW THEN PRINT:PRINT"Linha mui
to grande para centralizar ":PR
INTTT$:GOTO 3610
3520 SS=STRING$(INT((TW-LEN(TT$
))/2),32): IF P THEN LPRINT L1$,
GP$,SS$,TT$,L1$,GP$: ELSE PRINT
PRINTSS$,TT$
3530 PP=0:LC=LC+1:GOSUB 3630:GO
TO 3560
3540 IF P THEN LPRINT A$,MID$(T
T$,2): ELSE PRINTA$,MID$(TT$,2)
:
3550 PP=0:LC=LC+1:GOSUB 3630
3560 K=K+1: IF P=0 THEN FOR Z=1
TO 500:NEXT
3570 IF K<TL THEN 3290
3580 IF P THEN LPRINT LF$;LF$ E
LSE PRINT:PRINT
3590 IF KF=2 THEN CLOSE#1
3600 IF NOT P THEN 3190 ELSE RE
TURN
3610 PLAY "O3L3CR64L3CR64L8CR64
":FOR I=1 TO 2500:NEXT: IF KF
=2 THEN CLOSE#1
3620 RETURN
3630 IF LC>TH THEN IF P THEN LP
RINT LF$,LF$,GP$ ELSE PRINTLF$:
LF$:
3640 LC=1:RETURN
5000 I=CP:LOCATE 0,1:LINEINPUT
 "Procura por: ",TC$
5010 IF TC$="" THEN 5050
5020 PRINT"Procurando"
5030 IF I=TL THEN CP=TL:CLS:GOS
UB 2080:RETURN
5040 IF INSTR(TX$(I),TC$)=0 THE
N I=I+1:GOTO 5030
5050 CP=I+1:CLS:GOSUB 2080:RETU
RN
5060 IF SS>SE THEN SWAP SS,SE
5070 SE=SE-1
5080 LOCATE 0,20:PRINT"Ordenand
o"
5090 FOR I=SS TO SE-1
5100 K=I
5110 FOR J=I+1 TO SE
5120 IF TX$(J)<TX$(K) THEN K=J
5130 NEXT: IF I<>K THEN SWAP TX$
(K),TX$(I)
5140 NEXT:CLS:GOSUB 2080:RETURN
```

A rotina de ordenação é acessada a partir do modo editor. Leve o marcador para um dos extremos das linhas a serem ordenadas e tecle **CTRL-O**. A seguir, transfira o cursor para o outro extremo e tecle novamente **CTRL-O**. A ordenação se iniciará imediatamente.

A rotina para a procura de palavras também é ativada dentro do modo editor. Leve o marcador para a posição a partir da qual quer iniciar a busca. Tecle **CTRL-P** e digite a palavra que deve ser procurada. Tecle **[RETURN]** para iniciar a busca. Assim que a expressão for encontrada, o marcador será colocado abaixo da linha que a contém. Para continuar procurando outras ocorrências da mesma palavra, tecle **CRTL-P**; digite **[RETURN]** apenas quando a palavra for solicitada.

Para acessar a rotina de impressão, tecle I a partir do menu principal. Várias perguntas serão feitas. A primeira refere-se à possibilidade do texto a ser impresso vir da memória ou de um arquivo em fita.

A seguir, você deve responder se quer usar "máscara de texto" ou não. Essa rotina possibilita a inserção de partes variáveis no texto principal. Para isso, coloca-se pares de colchetes ("][") nas posições desejadas. Desse modo, quando o programa encontrar esse sinal no texto, irá substituí-lo por um bloco que pode ser carregado do teclado ou de um arquivo. Se você optar por dar entrada pelo teclado, deve digitar o texto necessário toda vez que for solicitado. Caso contrário, a informação será lida de um arquivo em fita (este deverá ter sido previamente preparado para isso).

Antes de iniciar a impressão, é possível também alterar a configuração da impressora. Os valores habituais são: 80 para a largura do formulário, 60 para a largura da linha de texto, 66 para o número total de linhas por formulário e 60 para o número de linhas de texto por página.

Os sinais para formatar o texto são nossos conhecidos: o asterisco ("*") centraliza o texto que o segue; o ampersand ("&") provoca um avanço de linha, de maneira que o texto subseqüente seja iniciado na primeira posição da linha seguinte; já o cifrão ("$"), além de fazer isso, provoca um recuo no texto; finalmente, o sustenido ("#") faz com que a linha seja impressa na margem direita da página.

Quando uma palavra não couber no trecho final de uma linha, deve-se evitar quebrá-la, para que ela não fique dividida por um espaço. O procedimento correto, neste caso, é digitar a palavra inteira na linha seguinte.



```
3000  HOME : BL$ = "IMPRIMIR": G
OSUB 250
3010  IF TL < 2 THEN 3060
3020  PRINT : PRINT : PRINT "IM
PRIME TEXTO DA [M]EMORIA OU [D]
ISCO? ": PRINT "TECLE <CR> PARA
 RETORNAR AO MENU ";
3030  GET R$: IF R$ = CHR$ (13
) THEN  RETURN
3040  IF R$ < > "M" AND R$ <
> "D" THEN 3030
3050  PRINT R$: IF R$ = "M" THE
N 3070
3060  GOSUB 4500: HOME : BL$ = "
IMPRIMIR": GOSUB 250
3070  IF TL = 1 THEN  PRINT : H
R$ (7): VTAB 12: HTAB 8: PRINT
 "NAO HA TEXTO NA MEMORIA": FO
R I = 1 TO 1000: NEXT : PRINT
CHR$ (7): RETURN
3080 KF = 0: HTAB 1: VTAB 9: PR
INT "QUER USAR MASCARA? ";
3090  GET R$: IF R$ < > "S" AN
D R$ < > "N" THEN 3090
3100  PRINT R$: IF R$ = "N" THE
N 3220
3110  PRINT : INPUT "QUANTOS BL
OCOS SERAO USADOS? (MAX 10) ";N
B: IF NB < 0 OR NB > 10 THEN 31
10
3120  PRINT : PRINT "OS BLOCOS
SERAO CARREGADOS DO": PRINT "[T
]ECLADO OU [D]ISCO? ";
3130  GET R$: IF R$ < > "T" AN
D R$ < > "D" THEN 3130
3140  PRINT R$:BL = 0:KF = 2: I
F R$ = "T" THEN KF = 1: GOTO 32
10
3150  PRINT : INPUT "NOME DO AR
QUIVO MASCARA? ";VB$
3160  IF  ASC (VB$) < 65 OR  AS
C (VB$) > 90 THEN 3150
3170 VB$ = VB$ + ".TXT": PRINT
: PRINT D$;"OPEN";VB$;",S";L1;"
,D",L2
```

```
3180 PRINT D$;"READ";VB$: INPU
T DV,DV
3190 IF DV < NB THEN PRINT D$
;"CLOSE":ER = 2: GOTO 3580
3200 FOR I = 1 TO NB: INPUT RP
$(I): NEXT : PRINT D$;"CLOSE":
GOTO 3220
3210 PRINT : PRINT : FOR I = 1
 TO NB: PRINT "BLOCO VARIAVEL "
;I,"=>";: INPUT RP$(I): NEXT
3220 HOME : GOSUB 250: VTAB 5:
 PRINT "QUER MUDAR A CONFIGURAC
AO DA IMPRESSORA? ";
3230 GET R$: IF R$ < > "S" AN
D R$ < > "N" THEN 3230
3240 PRINT R$: IF R$ = "S" THE
N GOSUB 5500: HOME :BL$ = "IMP
RIMIR": GOSUB 250
3250 VB = 0:PP = 0:AS = 0:LC =
1: PRINT : PRINT : PRINT "QUER
UMA DEMONSTRACAO NA TELA? ";
3260 GET R$: IF R$ < > "S" AN
D R$ < > "N" THEN 3260
3270 PRINT R$:P = 0:G1 = 0: IF
 R$ = "N" THEN P = 1:G1 = GP
3280 FOR K = 1 TO TL - 1: IF
LEFT$ (TX$(K),1) = "#" AND LEN
 (TX$(K)) - 1 > AS THEN AS = L
EN (TX$(K))
3290 NEXT : IF AS > TW THEN ER
 = 1: GOTO 3580
3300 K = 1: PRINT : PRINT D$;"P
R#";P: PRINT LF$: SPC( G1),:AS
 = "": IF AS > 0 THEN A1 = GP +
TW - AS
3310 TT$ = TX$(K)
3320 IF TT$ = " " THEN PRINT
 CHR$ (13); SPC( G1): PP = 0:LC
 = LC + 1: GOSUB 3640: GOTO 354
0
3330 IF KF = 0 THEN 3380
3340 FOR I = 1 TO LEN (TT$) -
 1: IF MID$ (TT$,I,2) < > "][
" THEN NEXT :BP = 0: GOTO 3380
3350 BP = I:BL = BL + 1
3360 IF BP = 1 THEN TT$ = RP$(
BL) + MID$ (TT$,3): GOTO 3380
3370 TT$ = LEFT$ (TT$,BP - 1)
+ RP$(BL) + MID$ (TT$,BP + 2):
 GOTO 3340
3380 TT = ASC (TT$): IF TT = 3
5 OR TT = 36 OR TT = 38 OR TT =
 42 THEN 3460
3390 IF PP + LEN (TT$) < = T
W THEN PRINT TT$; CHR$ (32);:P
P = PP + LEN (TT$) + 1: GOTO 3
540
3400 TA$ = LEFT$ (TT$,TW - PP)
3410 IF RIGHT$ (TA$,1) = " "
THEN 3440
3420 IF LEN (TA$) > 1 THEN TA
$ = LEFT$ (TA$, LEN (TA$) - 1)
: GOTO 3410
3430 TA$ = ""
3440 PRINT TA$, CHR$ (13); SPC
( G1);:PP = 0:LC = LC + 1: GOSU
B 3640:TT$ = MID$ (TT$, LEN (T
A$) + 1): IF TT$ = "" THEN 3540
3450 GOTO 3390
3460 FOR I = 1 TO 4: IF MID$
("&$*#",I,1) < > LEFT$ (TT$,1
) THEN NEXT
3470 ON I GOTO 3480,3490,3510,
3530
3480 PRINT CHR$ (13); SPC( G1
);:PP = 0:LC = LC + 1: GOSUB 36
40:TT$ = MID$ (TT$,2): GOTO 33
30
3490 TT$ = MID$ (TT$,2): PRINT
 CHR$ (13); SPC( G1); SPC( IN
T (TW / 4));:PP = INT (TW / 4)
3500 LC = LC + 1: GOSUB 3640: G
OTO 3330
3510 TT$ = MID$ (TT$,2): IF L
EN (TT$) > TW THEN ER = 3: GOTO
 3580
3520 PRINT CHR$ (13); SPC( G1
); SPC( INT ((TW - LEN (TT$))
 / 2));TT$; CHR$ (13); SPC( G1)
;:PP = 0:LC = LC + 1: GOSUB 364
0: GOTO 3540
3530 PRINT CHR$ (13); SPC( A1
); MID$ (TT$,2);:PP = 0:LC = LC
 + 1: GOSUB 3640: GOTO 3540
3540 K = K + 1: IF P = 0 THEN
FOR Z = 1 TO 1000: NEXT
3550 IF K < TL THEN 3310
3560 IF P = 1 THEN PRINT LF$;
LF$: PRINT D$,"PR#0": RETURN
3570 IF P = 0 THEN PRINT : PR
INT : GOTO 3250
3580 PRINT : PRINT D$;"PR#0":
PRINT CHR$ (7):
3590 HOME : VTAB 12: HTAB 10
ON ER GOTO 3600,3610,3620
3600 PRINT "ENDERECO MUITO LON
GO.": GOTO 3630
3610 PRINT "NAO HA BLOCOS SUFI
CIENTES": PRINT TAB( 12)"NO AR
QUIVO ESCOLHIDO!": GOTO 3630
3620 PRINT "IMPOSSIVEL CENTRAL
IZAR!"
3630 FOR I = 1 TO 2500: NEXT :
 PRINT CHR$ (7);: RETURN
3640 IF LC > TH THEN PRINT LF
$;LF$: SPC( G1);:LC = 1
3650 RETURN
5200 L = CP:TG$ = "": VTAB 20:
HTAB 1: PRINT "PROCURA POR: ";
5210 GET TC$: IF TC$ < > CHR
$ (13) THEN TG$ = TG$ + TC$: PR
INT TC$;: GOTO 5210
5220 IF TG$ = "" THEN 5270
5230 HTAB 1: PRINT "PROCURANDO
...";: CALL - 958
5240 IF L = TL THEN CP = TL: G
OTO 5270
5250 FOR I = 1 TO LEN (TX$(L)
) - 1: IF MID$ (TX$(L),I, LEN
(TG$)) < > TG$ THEN NEXT :L =
 L + 1: GOTO 5240
5260 CP = L + 1
5270 HOME : GOSUB 2090: RETURN
5300 IF SS > SE THEN TT = SS:S
S = SE:SE = TT
5310 SE = SE - 1
5320 HTAB 1: VTAB 20: PRINT "O
RDENANDO..."
5330 FOR I = SS TO SE - 1
5340 K = I
5350 FOR J = I + 1 TO SE
5360 IF TX$(J) < TX$(K) THEN K
 = J
5370 NEXT : IF I < > K THEN T
T$ = TX$(K):TX$(K) = TX$(I):TX$
(I) = TT$
5380 NEXT : HOME : GOSUB 2090:
RETURN
```

Para acessar as rotinas de ordenação e busca de palavras, você deve estar no modo editor.

Se você quiser ordenar um bloco de dados, leve o marcador ">" para um dos extremos do bloco e tecle **CTRL-O**. Em seguida, mova-o para o outro extremo e tecle novamente **CTRL-O**. A ordenação começará imediatamente.

Quando for preciso procurar uma determinada palavra dentro do texto, leve o marcador ao ponto escolhido. A seguir tecle **CTRL-P** e digite a palavra desejada, respondendo à solicitação. Tecle **[CR]** e a busca terá início. Quando a palavra for encontrada, o marcador será posicionado logo abaixo da linha que a contém. Para continuar procurando novas ocorrências dessa palavra, repita toda a operação.

Ao mesmo tempo, para imprimir o texto, tecle I a partir do menu principal. Em seguida, responda às perguntas que lhe são apresentadas.

A opção usar "máscara de texto" permite acrescentar ao texto principal blocos variáveis, com os quais podem ser produzidas, por exemplo, cartas individualizadas. Se você fizer essa opção, deixe nos locais apropriados do texto um sinal "][" para cada bloco. Esses sinais serão substituídos pelo texto adequado no momento da impressão.

O texto pode ser digitado a partir do teclado, ou lido de um arquivo em disco. No caso do teclado, o programa solicitará que você digite a frase à medida que os sinais sejam encontrados no texto. Se os blocos forem carregados do disco, o computador lerá um arquivo seqüencial que você deverá ter criado com os dados necessários, usando o próprio programa (se você quiser aproveitar um arquivo já existente, deve acrescentar ".TXT" ao nome dele). Cuide para que os dados estejam na ordem correta e para que sejam suficientes para todas as solicitações.

O texto pode ser formatado com os sinais habituais. (Eles devem sempre ser colocados na primeira posição da linha.) O asterisco ("*") centraliza o conteúdo da linha; o ampersand ("&") provoca um avanço de linha da impressora, colocando o texto subseqüente a partir do início da próxima linha; o cifrão ("$") tem a mesma função, com a diferença de que provoca um recuo no começo do texto; o sustenido ("#") faz as linhas que o contêm serem colocadas à margem direita do texto.

Evite inserir blocos variáveis nas linhas em que aparece o sustenido.

MSX: TECLAS PROGRAMÁVEIS

- O QUE SÃO TECLAS DE FUNÇÃO
- PROGRAMAÇÃO DAS TECLAS
- MONTAGEM DE UM MENU
- COMO DETECTAR INTERRUPÇÕES
- PROGRAME A TECLA <STOP>

O MSX oferece um recurso fantástico para dar aos programas um acabamento profissional: as teclas programáveis. Explorando-as bem, você conseguirá menus de funções instantâneos na tela.

A maioria dos computadores profissionais, como os cobiçados micros da linha PC, de dezesseis bits, tem teclados imensos, com cem teclas ou mais. Isso não só simplifica o acesso a um grande número de funções embutidas no sistema, pela pressão a uma única tecla, como permite que o usuário reprograme a função de parte das teclas, de acordo com suas necessidades.

Em número de dez a doze, as teclas programáveis pelo usuário, também chamadas de *teclas de função*, em geral se situam num bloco isolado, na parte



superior ou lateral do teclado. Essa disposição facilita a correspondência entre as teclas e os rótulos a elas atribuídos, cujo posicionamento pode ser exibido na última linha do vídeo, também sob controle de programa.

Os micros da linha MSX dispõem de cinco teclas programáveis, que estão localizadas na parte superior do teclado. Na realidade, elas possibilitam o acesso a dez funções, pois, pressionando-se a tecla <SHIFT> simultaneamente com uma tecla programável, ativa-se uma segunda função que pode ser atribuída àquela tecla.

Os recursos de programação disponíveis no interpretator BASIC do MSX são muito ricos. Aprendendo a explorá-los de maneira criativa, você se habilitará a produzir programas com um desempenho altamente profissional. Neste artigo, apresentamos alguns truques interessantes, muitos dos quais não se encontram no Manual de BASIC que acompanha o computador. Você logo será capaz de aproveitá-los com sucesso em outros programas.

TECLAS PROGRAMÁVEIS

As teclas de função podem ser usadas de dois modos em um programa:

— para dar entrada a uma cadeia de caracteres no computador, pressionando-se uma única tecla;

— para interromper automaticamente a execução de um programa e redirecioná-lo para executar uma ou mais sub-rotinas específicas, cada qual associada a uma tecla programável.

Em ambos os casos, é possível (mas não obrigatório) exibir em uma linha reservada (a última linha da tela) uma lista de rótulos de identificação das teclas. Assim, o usuário não precisará recorrer sempre à memória para saber o que faz cada tecla programável. Essa linha, uma vez "ligada", fica presente na tela, mesmo quando seu conteúdo "rola".

COMANDOS INSTANTÂNEOS

O modo mais simples de utilização das teclas programáveis, que consiste em dar entrada a uma cadeia de caracteres, é exemplificado pela operação normal do próprio computador.

Como você já deve ter observado, ao ligar a máquina (ou ao pressionar o botão <**RESET**> de inicialização, existente em alguns micros da linha MSX), aparece uma série de rótulos na linha inferior da tela. Eles correspondem a comandos ou instruções do BASIC, ou seja, **COLOR**, **LIST**, **RUN**, **GOTO** etc.

Se você pressionar uma das teclas programáveis (identificadas na parte superior do teclado pelos rótulos **F1**, **F2** etc.), verá que a palavra a ela associada é entrada por extenso no computador, aparecendo na tela logo adiante do cursor.

Caso pressione a tecla <**SHIFT**> o conteúdo da linha inferior da tela mudará para outros quatro rótulos adicionais, que correspondem às funções das teclas **F6** a **F10**. Portanto, a pressão simultânea das teclas **F1** e <SHIFT> provocará a entrada automática da cadeia associada.

O carregamento dos rótulos associados a cada tecla é feito automaticamente, no momento de inicialização do interpretador BASIC. Para apagar a linha da tela, basta dar o comando:

```
KEY OFF
```

seja em modo direto, seja dentro de um programa.

Tente fazer isso e, em seguida, pressione novamente uma das teclas de função. Você verá que ela continua funcionando — a cadeia de caracteres a que se associa aparece do mesmo jeito na tela. A função do comando **KEY OFF**, portanto, resume-se ao desligamento da linha de exibição.

O comando seguinte faz exatamente o contrário do anterior, ou seja, torna a exibir a linha de tela associada às teclas programáveis:

```
KEY ON
```

Este comando também pode ser digitado em modo direto ou ser colocado dentro de um programa.

No caso da programação BASIC, a vantagem de se associar os comandos mais comuns às teclas programáveis é evidente: em vez de digitar o comando por extenso, pressiona-se a tecla programável uma vez. Com isso, evitamos erros de digitação e aceleramos consideravelmente o processo de entrada de um programa BASIC.

Experimentando todas as teclas programáveis, você observará que algumas delas, ao serem pressionadas, simplesmente dão entrada à cadeia de caracteres (por exemplo, **LIST**), e mais nada acontece. O cursor fica parado no fim da palavra entrada. Para que o comando seja executado, você deverá pressionar a tecla <**ENTER**>, que o envia para o processador. No exemplo dado, a execução resultaria na listagem de um programa inteiro na tela.

Você pode, porém, digitar o número ou os números de linha sobre os quais o comando **LIST** vai atuar, e só depois pressionar <**ENTER**>.

Algumas teclas programáveis, por outro lado, dão entrada à palavra associada, mas a executam imediatamente — é o caso de **RUN**. Como veremos adiante, essa diferença de funcionamento pode ser especificada facilmente pelo programador. Tudo depende da função que se quer associar a cada tecla.

Para verificar na tela quais os rótulos que se associam a cada uma das teclas programáveis, digite o comando **KEY LIST**. As palavras que são enviadas imediatamente, sem a necessidade do <**ENTER**>, aparecem assinaladas por uma flechinha no final. Sabe-se, desse modo, que o caractere de controle equivalente ao <**ENTER**> foi agregado ao final da palavra.

PROGRAMAÇÃO DAS TECLAS

É muito fácil atribuir uma cadeia de caracteres a uma tecla programável, e pode-se fazê-lo tanto em modo direto quanto dentro de um programa. A instrução em BASIC a ser utilizada é:

```
KEY n,"cadeia"
```

onde **n** é o número da tecla de função, e "**cadeia**", o rótulo que deve ser associado à tecla. Experimente digitar, por exemplo:

```
KEY 1,"CLS"
```

O rótulo da tecla **F1**, que inicialmente era **COLOR**, passa a ser **CLS**, ao ser inicializado o computador. Confira, digitando **KEY LIST**.

Agora, pressione a tecla **F1**. Você verá então que o comando **CLS** aparece instantaneamente na tela, logo após o cursor, mas não é executado. Para que isso ocorra, você deverá pressionar a tecla <**ENTER**>.

Se quisermos que a execução do comando seja automática, precisaremos definir a tecla de outro modo:

```
KEY 1,"CLS"+CHR$(13)
```

Ao listar o conteúdo das teclas programáveis com **KEY LIST**, você constatará que a marca de retorno foi acrescentada no fim da palavra **CLS**.

O caractere ASCII 13, adicionado à cadeia de caracteres por meio da função **CHR$**, corresponde exatamente ao código gerado internamente ao se pressionar a tecla <**ENTER**>. Ou seja, equivale a pressioná-la automaticamente, quando a tecla **F1** é acionada.

Cabem exatamente seis caracteres no espaço alocado para o rótulo de cada te-



cla, na linha de menu. Isto é o que cabe na tela, o que não significa que você esteja limitado a definir um rótulo de apenas seis caracteres. Experimente digitar:

```
KEY 1,"DEFUSR1=31100"+CHR$(13)
```

Apenas as seis primeiras letras (**DEFUSR**) aparecerão na tela, mas a cadeia será enviada por extenso quando se pressionar a tecla **F1**. Portanto, se for necessário "esconder" a marca de retorno em um rótulo, basta digitar a seguinte linha:

```
KEY 1,"CLS   "+CHR$(13)
```

Deixe três espaços entre o final da palavra e o segundo sinal de aspas, nesse exemplo.

Se a cadeia de caracteres a ser associada à tecla tiver que conter aspas em seu interior, não dará certo fazer algo como:

```
KEY 1,"LOAD "" CODE"
```

A função **CHR$**, mais uma vez, resolve o dilema.

```
KEY 1,"LOAD "+CHR$(44)+
CHR$(44)+" CODE"
```

O código ASCII 44, evidentemente, corresponde ao caractere aspas.

OUTROS USOS

Não é obrigatório atribuir às teclas programáveis apenas comandos ou instruções em BASIC. Na realidade, pode-se atribuir a elas qualquer cadeia de caracteres, o que possibilita muitas aplicações interessantes.

Suponhamos, por exemplo, que você vá passar uma longa mensagem ao computador usando um programa de processamento de textos. Se notar que certas palavras ou frases mais longas serão empregadas várias vezes (como **microcomputador**, **Secretaria Especial de Informática** etc.), bastará carregar previamente as teclas de função com estas cadeias (não seguidas do caractere ASCII 13). Pelas seis primeiras letras da frase, você poderá identificar a tecla associada a ela e, simplesmente, pressioná-la. Isso poupará muito tempo de digitação!

TÉCNICAS DE PROGRAMAÇÃO DE MENUS

Como já foi mencionado, as teclas programáveis são muito utilizadas para a programação rápida de menus, oferecendo duas vantagens: o menu fica sempre presente na linha reservada da tabela, e é muito mais fácil localizar as teclas associadas às suas opções do que se estivéssemos utilizando teclas normais do teclado.

Os programas que se seguem vão mostrar algumas das técnicas mais comuns de programação de menus utilizando as teclas de função.

Suponhamos que você queira elaborar um programa de banco de dados, com as quatro funções clássicas: inserir registros, apagar registros, modificar registros e listar o banco de dados. Além dessas funções precisaríamos de uma quinta opção: a de término da execução do programa.

Já vimos como montar um menu desse tipo de diversas maneiras — usando **PRINT**, **INPUT**, **INKEY$** etc. Para usar teclas programáveis, poderíamos fazer o seguinte:

```
10 ST$=STRING$(38,"-")
20 KEY OFF:CLS
30 KEY 1,"Insere"
40 KEY 2,"Apaga "
50 KEY 3,"Modif "
60 KEY 4,"Lista "
70 KEY 5,"FIM   "
80 LOCATE 10,0
90 PRINT "BANCO DE DADOS"
100 LOCATE 0,1:PRINT ST$
110 LOCATE 0,21:PRINT ST$
120 KEY ON
130 A$=INPUT$ (6)
140 LOCATE 0,10
150 PRINT "FUNÇAO: ";A$
160 IF A$="FIM   " THEN CLS:END
170 GOTO 130
```

A linha 10 define uma variável, **ST$**, igual a uma cadeia de 38 tracinhos: é a linha de separação, usada para dar melhor aparência à tela de entrada. A linha 20 "desliga" a linha de identificação das teclas programáveis e limpa a tela. As linhas 30 a 70 programam as teclas de função **F1** a **F5**. As linhas de 90 a 110 compõem a tela e a 120 "liga" novamente a linha de exibição dos rótulos das teclas de função, desta vez com os rótulos que foram atribuídos no programa.

Finalmente, as linhas 130 a 150 processam o resultado da pressão às teclas de função. Observe que a instrução

INPUT$, própria do MSX, tem um funcionamento semelhante ao do INKEY$, só que inclui um determinado número de pressões às teclas, de cada vez. INPUT$(6) significa que o computador deve esperar que seis caracteres sejam digitados (não os mostra na tela) para, então, prosseguir o programa.

Note que colocamos seis caracteres em cada rótulo, preenchendo-os com espaços em branco ao final, quando necessário. Assim, qualquer pressão a uma tecla programável imediatamente atenderá à condição da linha 130, armazenando o resultado na variável A$ e seguindo para as linhas 140 a 160.

Às linhas 140 e 150 cabe simplesmente escrever o resultado dessa atuação no meio da tela. A linha 160 exemplifica o que fazer a partir daí. Ela verifica se FIM foi pressionado e executa CLS:END, terminando o programa, se isto aconteceu.

Diversos IF seriam utilizados para chamar (GOSUB) as rotinas de inserção, apagamento etc., no nosso banco de dados. A linha 170 faz tudo voltar ao menu. Pode ser interessante dar um KEY OFF antes de chamar as rotinas, para que o usuário não tenha a impressão de que o menu ainda continua disponível. Isso pode ser feito seletivamente, como veremos mais adiante.

O programa seguinte, que é uma modificação do anterior, testa o resultado da pressão a uma tecla de função com mais economia

```
10 ST$=STRING$(38,"_")
20 KEY OFF:CLS
30 KEY 1,"Insere"
40 KEY 2,"Apaga "
50 KEY 3,"Modif "
60 KEY 4,"Lista "
70 KEY 5,"FIM   "
80 LOCATE 10,0
90 PRINT "BANCO DE DADOS"
100 LOCATE 0,1:PRINT ST$
110 LOCATE 0,21 PRINT ST$
120 KEY ON
130 A$=LEFT$(INPUT$(6),1)
140 LOCATE 0,10
150 ON INSTR("IAMLF",A$) GOSUB
170,180,190,200,900
155 GOTO 130
170 PRINT "ROTINA DE INSERÇAO
 "
175 RETURN
180 PRINT "ROTINA DE APAGAMENTO
 "
185 RETURN
190 PRINT "ROTINA DE MODIFICAÇA
O"
195 RETURN
200 PRINT "ROTINA DE LISTAGEM
 "
205 RETURN
900 REM FIM DO PROGRAMA
905 CLS:END
```

Observe que, agora, a linha 130 é usada para extrair e armazenar em A$ apenas o primeiro caractere do rótulo da tecla de função acionada. Portanto, os rótulos precisam ter, como no exemplo, letras iniciais diferentes, para evitar confusões.

Na linha 15 está a vantagem dessa técnica. Ela permite utilizar a forma bem mais compacta do ON...GOSUB, para dirigir o programa à rotina chamada pelo menu.

A função INSTR, que talvez você não conheça, significa, em inglês, *in string*, ou seja, *dentro de um cordão*. Ela verifica se um caractere ou cadeia de caracteres está presente dentro de uma outra cadeia. Em caso negativo, o valor zero é retornado. Em caso afirmativo, a posição do caractere (sua ordem no primeiro cordão) é retornada. Por exemplo:

```
PRINT INSTR ("COMPUTADOR","P")
```

retorna o valor 4, enquanto:

```
LET A$="X"
PRINT INSTR ("COMPUTADOR",A$)
```

retorna o valor zero.

No programa, o cordão IAMLF especificado corresponde às letras iniciais das cinco opções oferecidas pelo menu. Sendo pressionada uma das teclas programáveis, a letra inicial do rótulo é armazenada em A$ (linha 130) e testada pela função INSTR, na linha 150. O número retornado serve para endereçar a sub-rotina que será selecionada em ON...GOSUB.

A listagem do programa anterior mostra apenas a primeira e a última linhas das rotinas. Se estiver interessado, complete-as você mesmo.

INTERRUPÇÕES PROGRAMADAS

Os programas vistos até aqui parecem não apresentar vantagens mais significativas em relação à programação convencional de menus que aprendemos anteriormente. Na realidade, eles apenas acrescentam a possibilidade de utilização das teclas programáveis e da linha reservada de rotulação.

Entretanto, as teclas de função do micro MSX oferecem um recurso bem mais poderoso de programação: a *interrupção programada*.

Quando pressionamos uma das teclas de função F1 a F10, o comportamento do computador é diferente do observado em relação às teclas normais do teclado. Em vez de simplesmente gerar um caractere ou cordão de caracteres, a pressão à tecla de função "avisa" o interpretador BASIC — qualquer que seja o ponto do programa que esteja sendo executado — que tal pressão ocorreu, gerando o que se denomina, em linguagem técnica, de interrupção do processador (UCP).

Para entender como funciona uma interrupção pelo teclado, basta tomar como exemplo uma tecla que já é pré-programada para gerar um tipo específico de interrupção: a tecla <STOP>.

O sistema operacional e interpretador BASIC do MSX está sempre "atento" à tecla <STOP>, por meio de um dispositivo especial de hardware, para detectar se ela foi pressionada. A verificação é feita a cada poucos microssegundos e, não importa o que o computador esteja fazendo no momento da interrupção, imediatamente o processador passa a executar uma rotina de serviço que, no caso, servirá para "congelar" um programa em execução ou interrompê-lo, retornando ao modo de entrada (quando <CONTROL> e <STOP> são pressionadas simultaneamente).

Pois bem, F1 a F10 são teclas de interrupção *programáveis*, ou seja, o programador pode especificar a rotina de serviço que deve ser acionada pelo sistema operacional correspondente a cada tecla.

Existem diversas instruções do BASIC que permitem programar interrupções a partir das teclas F1 a F10. Uma delas consiste em:

```
ON KEY GOSUB n1,n2...
```

que verifica se alguma tecla de função foi pressionada, ou seja, se ocorreu uma interrupção. Conforme a tecla pressionada, um número inteiro entre 1 e 10 é retornado. Ele chama a sub-rotina na ordem indicada, nas linhas n1, n2 etc. ON KEY GOSUB 1000, 1100, por exemplo, provocará um salto para a sub-rotina que começa na linha 1000, se a tecla F1 for acionada, ou para a linha 1100, se a tecla F2 for acionada. A instrução ON KEY GOSUB deve ser colocada em um ponto da listagem que assegure a sua execução pelo menos uma vez, antes das detecções de interrupção, que, como a rotulação das teclas, ocorrem no começo do programa.

Porém, precisamos de mais uma condição para montar a "armadilha" que detecta interrupções. A capacidade de gerar interrupções pode ser "ligada" ou "desligada" de cada tecla de função individualmente, pelas instruções:

```
KEY (n) ON

KEY (n) OFF

KEY (n) STOP
```

onde n é o número da tecla. Quando se liga o computador, todas as teclas estão em modo **OFF**, ou seja, "desligadas". Ao ser detectada uma pressão à tecla de função, um **KEY STOP** é dado automaticamente, inibindo toda a interrupção subsequente, até que ocorra um **RETURN** da sub-rotina chamada pelo **ON KEY GOSUB**

Embora o comando **KEY (n) STOP** iniba a interrupção do programa, ele continua sendo capaz de detectar uma pressão à tecla de função indicada. Assim, ao retornar da rotina de serviço de uma tecla de função, o programa poderá imediatamente rumar para outra rotina, se, enquanto isso, outra tecla de função tiver sido pressionada.

Se o programador quiser inibir tanto a interrupção quanto a detecção, deve usar um comando **KEY (n) OFF**.

No programa seguinte você encontrará exemplos das técnicas mais elementares de utilização das instruções **ON KEY GOSUB** e **KEY ON**. O objetivo é imitar uma maquina de somar simples, que apresenta o resultado quando a tecla **F1** é pressionada.

```
10 CLS
20 PRINT TAB(10),"SOMADORA MSX
I"
30 ON KEY GOSUB 200
40 S=0
50 KEY (1) ON:KEY 1,"SOMA  "
60 KEY 2," ".KEY 3," "   KEY
4," ":KEY 5," "
120 '
130 '    ALÇA DE SOMA
140
150 LOCATE 6,10
160 PRINT "                          "
170 LOCATE 0,10
180 INPUT "VALOR ";V
190 S=S+V GOTO 150
200 '
210 '    SUBROTINA PARA TECLA F1
220 '
230 LOCATE 0,12
240 PRINT "SOMA = ";S
250 RETURN 150
```

A linha 30, logo no começo do programa, prepara a armadilha para detectar interrupções pelas teclas de função e desviar o fluxo de processamento. A linha 40 zera a variável S, que é o acumulador de somas; a linha 50 "liga" a tecla **F1** e rotula seu propósito na linha



reservada para exibição na tela. A linha 60 "limpa" os rótulos das teclas programáveis que não serão usadas.

As linhas 150 a 190 do programa realizam a função de soma de valores. Elas formam uma alça sem fim, o que é típico de todo programa que utiliza as teclas de função do contexto do **ON KEY GOSUB**. A alça sem fim faz o programa esperar ou executar alguma coisa (no caso somar os números entrados pelo usuário), até que uma tecla de interrupção seja pressionada.

Se se pressionar a tecla **F1** enquanto o programa está esperando a entrada de um valor, nada acontece de imediato. Mas, assim que a tecla **<ENTER>** for pressionada, a rotina que começa na linha 200 será acionada, pois foi a especificada na linha 30. Ela simplesmente mostra a soma acumulada até o último número entrado e retorna para a linha 30, de modo a dar prosseguimento à soma.

E o que acontece se alguma tecla programável não desativada for acidentalmente pressionada? Não acontece nada, pois o funcionamento do **ON KEY GOSUB** é tal que o desvio se realiza apenas para as sub-rotinas indicadas.

UM PROGRAMA MAIS FUNCIONAL

Embora não apresente problemas de funcionamento, o programa anterior tem dois defeitos: não oferece provisão para interrupção ao final do processamento, nem a possibilidade de zerar o acumulador para permitir nova soma sem sair do programa. O seguinte, mais completo, faz tudo isso:

```
10 CLS
20 PRINT TAB(10);"SOMADORA MSX
II"
30 ON KEY GOSUB 200,300,400
40 S=0
50 KEY (1) ON:KEY 1,"SOMA   "
60 KEY (2) ON:KEY 2,"LIMPA "
70 KEY (3) ON:KEY 3,"FIM    "
80 KEY 4," ":KEY 5," "
120 '
130 '    ALÇA DE SOMA
140 '
150 LOCATE 6,10
160 PRINT "                        "
170 LOCATE 0,10
180 INPUT "VALOR ",V
190 S=S+V:GOTO 150
200 '
210 '    SUBROTINA PARA TECLA F1
220 '
230 LOCATE 0,12
240 PRINT "SOMA = ";S
250 RETURN 150
300 '
310 '    SUBROTINA PARA TECLA F2
320 '
330 LOCATE 0,12
340 PRINT "SOMA = 0                  "
350 S=0:RETURN 150
400 '
410       SUBROTINA PARA TECLA F3
420 '
430 CLS:END
```

Agora, a linha 30 permite o desvio para três rotinas: de soma (linha 200), de zeragem (linha 300) e de término (linha 400). Experimente pressionar outras teclas de função enquanto uma rotina está sendo executada: se você conseguir fazê-lo a tempo, notará que nada acontece.

Em um programa curto como este, a definição individual de cada **KEY** é mais do que satisfatória. Entretanto, se precisar incluir muitas teclas, você poderá obter um programa mais compacto e elegante fazendo esta definição dentro de um laço de repetição. Para ter um exemplo, substitua estas linhas no programa anterior:

```
50 FOR I=1 TO 5
60 READ C$:KEY (I) ON
70 KEY I,C$:NEXT I
80 DATA "SOMA  ","LIMPA ","FIM
"
```

O laço que vai das linhas 50 a 70 faz um índice **I** variar de 1 a 5. A linha 60 lê sucessivamente os rótulos das teclas, em **DATA**. Observe que os rótulos não usados são definidos como um espaço em branco.

As linhas 60 e 70 ainda se encarregam, respectivamente, de "ligar" a tecla e atribuir-lhe o rótulo **C$**. Note que os comandos aceitam variáveis como argumentos.

Ao experimentar o programa, você deve ter reparado que as teclas programáveis não interrompem o programa enquanto ele está esperando entradas pelo usuário, em um comando **INPUT**. Como se trata de um inconveniente, recorreremos a um expediente que possibilitará a ocorrência de interrupção em qualquer momento. A condição para isso é que o computador esteja dentro de um laço de repetição durante a entrada de um valor na linha 30. Tal condição pode ser atendida se "simularmos" um comando **INPUT**, por meio de vários **INKEY$** sucessivos.

A rotina seguinte, que começa na linha 500, faz esse serviço, e retorna um valor númerico **V**. Substitua também a linha 30, como foi indicado:

```
180 PRINT "VALOR ";:GOSUB 500
500 V$=""
510 C$=INKEY$:IF C$="" THEN 510
520 IF ASC(C$)=13 THEN V=VAL(
V$):RETURN
530 PRINT C$;:V$=V$+C$:GOTO 510
```

Experimente o programa modificado e veja como ele é capaz de aceitar interrupções também dentro do ciclo de entrada de dados.

PROGRAMAÇÃO DA TECLA <STOP>

Afirmamos um pouco antes que **<STOP>** é uma tecla de interrupção pré-programada, ou seja, que executa uma rotina fixa ao ser acionada. Bem, a afirmação é totalmente verdadeira!

De fato, o usuário também pode programar a tecla **<STOP>**, definindo o que ela deve fazer quando for pressionada. Assim, **<STOP>** equivale às teclas de função, só que não dispõe de espaço próprio na linha reservada de rotulação, para dizer ao usuário o que ela faz. Mas podemos utilizar a própria definição da tecla e usá-la sempre para terminar um programa ou um subprograma. Esta função de "escape" de um nível de programa para outro é a marca registrada dos superprogramas comerciais atualmente disponíveis para computadores pessoais.

O comando que se segue desvia o processamento para a sub-rotina indicada, quando as teclas **<CTRL>** e **<STOP>** são pressionadas simultaneamente (nada acontece se se pressionar apenas a tecla **<STOP>**).

```
ON STOP GOSUB
```

Para que uma instrução **ON STOP GOSUB** funcione, é necessário ativar a armadilha interna, por meio de um comando **STOP ON**. A partir do momento em que este comando for encontrado, o desvio será realizado para a sub-rotina indicada no **ON STOP GOSUB**, sempre que se pressionar simultaneamente as teclas **<CTRL>** e **<STOP>**. Como nas teclas de função, os comandos **STOP OFF** e **STOP STOP** "desligam", respectivamente, a interrupção e a detecção, para as teclas mencionadas. Experimente alterar o programa, adicionando a opção de interrupção pelas teclas **<CTRL> <STOP>**:

```
45 ON STOP GOSUB 400
90 STOP ON
```

A possibilidade de detectar o acionamento da tecla **<STOP>** tem uma outra utilidade: muitas vezes, o programador deseja impedir a interrupção acidental ou intencional de seu programa por um outro usuário, seja para protegê-lo contra olhos curiosos, seja para evitar perdas de dados ou outras condições críticas. O **ON STOP GOSUB** funciona muito bem para esta finalidade, no micro MSX.

SPRITES PARA O TRS-80 (1)

- SÍMBOLOS GRÁFICOS
- UTILIZAÇÃO DE CARACTERES GRÁFICOS EM PROGRAMAS
- AS FUNÇÕES CHR$ E STRING$
- TABELA DE REFERÊNCIA

Os usuários do TRS-80 não precisam ficar frustrados por não disporem dos fabulosos sprites. Como verão aqui, é possível obter animações de excelente qualidade em suas máquinas.

Se você já tentou produzir gráficos animados em um microcomputador da linha TRS-80 usando os comandos **SET** e **RESET**, com certeza chegou à conclusão de que a tarefa é praticamente impossível. Esses comandos servem para acender ou apagar pixels individuais na tabela de baixa resolução, permitindo compor, sem maiores dificuldades, figuras complexas, como um avião ou um foguete. Entretanto, se tentarmos animar o desenho como um todo — deslocá-lo na tela, por exemplo —, obteremos efeitos ridículos, graças à lentidão do interpretador BASIC.

Vimos em artigos anteriores que, no que se refere aos micros pessoais mais modernos — como os das linhas Spectrum, TRS-Color, MSX e Apple —, a solução para esse problema consiste em definir caracteres gráficos por software, armazená-los em uma porção da memória (área UDG no Spectrum, sprites autênticos no MSX e variáveis numéricas ou alfanuméricas nos demais computadores) e animá-los.

Evidentemente, o TRS-80 não conta com recursos tão maravilhosos. Mas há um "jeitinho" que nos permite obter animações fantasticamente rápidas... em BASIC! Veremos como em uma série de artigos. Antes, porém, precisamos entender o funcionamento dos caracteres gráficos pré-definidos do TRS-80.

CARACTERES GRÁFICOS

O que são caracteres gráficos? Como você já sabe, os caracteres que aparecem no vídeo têm códigos numéricos inteiros que, teoricamente, podem variar entre 0 e 255 — cada caractere correspondendo, portanto, a um byte da memória de vídeo. Parte dessa codificação é convencionada internacionalmente: trata-se do chamado código ASCII, que vai de 32 a 126. Os códigos 0 a 31 são normalmente utilizados para funções de controle do vídeo, e dependem do tipo de computador. O mesmo acontece com os códigos 127 a 255. Nesta faixa cada fabricante utiliza os códigos de uma maneira, geralmente para acomodar caracteres gráficos.

O TRS-80 possui 64 caracteres gráficos de teclado, ocupando a faixa de códigos que vai de 128 a 191.

O caractere gráfico com o código 128 é um espaço em branco, semelhante ao que recebe o código 32 no ASCII. Entretanto, não deixa de ser útil, sobretudo para efeito de certas manipulações nas telas gráficas.

Um aspecto bastante interessante dos caracteres gráficos pré-definidos é que eles se comportam exatamente como qualquer outro caractere alfanumérico, para fins de impressão na tela. Assim, eles podem ser usados como argumento de comandos **PRINT**, o que nos permite obter boa velocidade na exibição de gráficos no micro TRS-80.

Ao contrário dos caracteres gráficos pré-definidos do TK-2000, do MSX, do ZX-81 e do Spectrum, os do TRS-80 não podem ser entrados pelo teclado. Assim, para especificá-los dentro de um programa, é necessário utilizar as funções **CHR$** e **STRING$**

Para imprimir na tela um retângulo totalmente preenchido, escrevemos, por exemplo:

```
PRINT CHR$(191)
```

O programa abaixo mostra a tabela de correspondência entre códigos numéricos e gráficos na tela do TRS-80:

```
10 CLS
20 FOR J=129 TO 191 STEP 9
30 FOR I=J TO J+8
40 PRINT I;CHR$(I);
50 NEXT I·PRINT NEXT J
```

O programa funciona da seguinte maneira: a linha 10 limpa a tela. O laço que vai da linha 20 à linha 50 (controlado pela variável **J**) produz valores numéricos que variam entre 129 e 191 (a faixa de códigos gráficos, portanto), em incrementos de 9. Assim, o laço das linhas 30 a 50 (controlado pela variável **I**) pode incrementar de 1 em 1 os valores intermediários. Este é um artifício utilizado para produzir na tela uma tabela clara e bem-feita, com nove códigos por linha.

A linha 40 do programa exibe o código numérico (**I**) e o caractere gráfico correspondente (**CHR$(I)**). O **PRINT** entre os dois **NEXT** na linha 50 serve para mudar a linha de impressão.

Se você pretende empregar caracteres gráficos frequentemente em um programa, convém armazenar os seus códigos em uma variável alfanumérica. Com esse procedimento, você não só poderá utilizá-los com mais facilidade como também não precisará consultar a toda hora a tabela de códigos gráficos, ao desenvolver um programa.

Da mesma maneira, se você quiser utilizar uma cadeia de caracteres gráficos em vários pontos de um programa, armazene-a em uma variável alfanumérica. Para isso, recorra à função **STRING$**.

Execute o programa abaixo e veja como ficam linhas compostas pelo mesmo caractere gráfico.

```
10 CLS
20 INPUT"CODIGO GRAFICO
(129 191)";C
30 INPUT"COMPRIMENTO (1-63)":N
40 S$=STRING$(N.C)
50 PRINT @ 256,S$
60 FOR I=1 TO 500·NEXT
70 GOTO 10
```

A função **STRING$** tem várias utilidades, quando empregada com códigos gráficos. Uma delas, por exemplo, é preencher instantaneamente a tela com um caractere gráfico. Lembre-se de que a tela de textos do TRS-80 tem 1024 caracteres (dezesseis linhas de 64 colunas cada). Se usarmos, por exemplo, uma linha de programa composta de quatro **PRINT STRING$(256,191)**, separados por ponto e vírgula, o resultado será um preenchimento total da tela com o caractere gráfico 191, que corresponde a um bloco inteiramente preenchido (veja a tabela constante do seu Manual de Programação).

Substituindo o segundo argumento da função **STRING$** por outro código qualquer, poderemos usar o mesmo recurso para preencher a tela com blocos gráficos diversos.

PROGRAMAÇÃO DE GRÁFICOS EM 3-D (2)

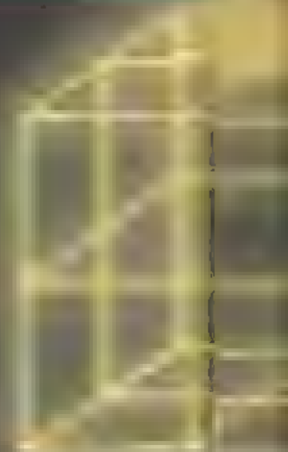
Até aqui, limitamos nossos [illegible] a linhas e formas bidimensionais. Mas, com algumas alterações nas rotinas utilizadas anteriormente, poderemos construir imagens tridimensionais.

Na teoria, pelo menos, não faz sentido falar na criação de um desenho tridimensional — uma caixa, por exemplo — num plano bidimensional, seja ele um pedaço de papel ou uma tela de televisão. No entanto, podemos recorrer a técnicas de desenho que dão às figuras a impressão de solidez.

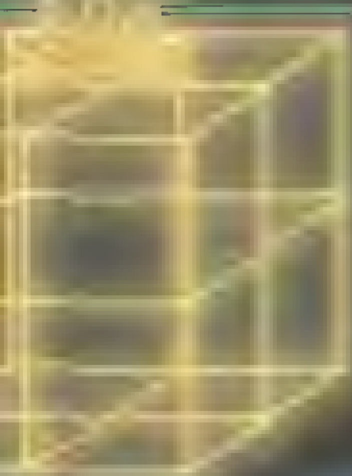
PERSPECTIVA E PROJEÇÃO ISOMÉTRICA

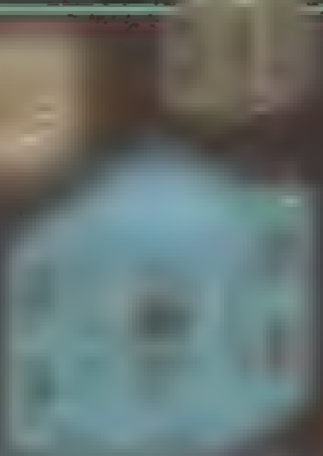
A pintura de uma paisagem, por exemplo, mostra profundidade [illegible] perspectiva. Trata-se de um truque visual baseado no fato de que os objetos parecem ficar menores à medida que se distanciam, e de que linhas paralelas parecem convergir à distância.

A perspectiva não é a única [illegible] usada nesse tipo de representação. Em desenho técnico é comum ouvirmos falar de projeção isométrica. Como no desenho em perspectiva, as linhas que o observador não vê são desenhadas num certo ângulo. [illegible]



- DESENHANDO EM TRÊS DIMENSÕES
- PROJEÇÕES ISOMÉTRICAS
- MUDANÇA DE COORDENADAS
- MONTAGEM DE UM CUBO COM GRADES
- COMO UTILIZAR O PROGRAMA PARA OBTER OUTRAS FORMAS
- UM LAÇO NA CAIXA









[illegible] à distância. A vantagem da projeção isométrica é que a direção da linha não afeta a escala e, por isso, pode-se saber a medida exata de cada linha diretamente do desenho.

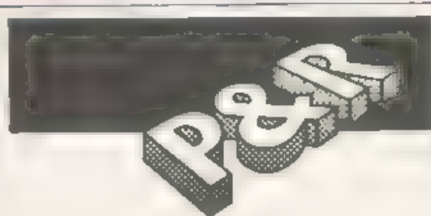
Embora possamos utilizar o computador para fazer um desenho em perspectiva, é bem mais fácil representar a projeção isométrica. Esta baseia-se simplesmente na criação de um terceiro eixo, além do eixo X (horizontal) e do eixo Y (vertical) já conhecidos. Este terceiro eixo, o Z, está posicionado num certo ângulo com os outros dois. Qualquer linha desenhada nesse ângulo pode ser entendida como se afastando (ou se aproximando) do observador. O diagrama dos eixos na figura da página 630 ajudará a entender melhor o que dissemos até agora.

ALTERAÇÕES NO PROGRAMA

Recorrendo à projeção isométrica, podemos produzir imagens tridimensionais tipo wireframe a partir de formas bidimensionais, utilizando as rotinas para grades e círculos apresentadas no primeiro artigo da série. Algumas dessas formas bidimensionais precisarão ser distorcidas, de modo que se tornem compatíveis com o ângulo do eixo Z. Para isso procederemos a várias mudanças de coordenadas baseadas em princípios matemáticos um tanto complicados. Mas não se preocupe, pois o computador as manuseia com facilidade.

Em primeiro lugar, transformaremos coordenadas 2-D (X,Y) em 3-D (X', Y', Z'), no plano desejado. Isto significa que devemos especificar o quanto o plano se estende ao longo do eixo Z. Depois transformaremos as coordenadas 3-D (X', Y', Z') num novo conjunto de coordenadas 3-D (Xe,Ye,Ze), baseado na direção e posição da qual olhamos o objeto. Finalmente, voltaremos a transformar (Xe,Ye,Ze) em coordenadas 2-D (Xp,Yp), para que o conjunto possa ser mostrado na tela. Acompanhe os programas da página 631; eles mostram a modificação de uma forma de desenho nos seis lados de uma imagem.

Durante a transformação, podemos levar em consideração a perspectiva,



Posso modificar o programa para desenhar imagens diferentes?

A rotina base de nosso programa é a que compõe a grade. A rotina que ela utiliza para traçar linhas poderia ser modificada para desenhar outras formas. No caso de um triângulo, por exemplo, não haverá maiores problemas, mas, se tivermos que acessar a rotina várias vezes seguidas, tal como no desenho de um tetraedro (figura de quatro lados triangulares), precisaremos de um pouco mais de atenção. No desenho da base, os valores dados às variáveis de transformação serão iguais aos da caixa. Em compensação, a alteração das variáveis de transformação (linhas 270 a 300) para obter o ângulo e posição correta dos outros lados constitui uma tarefa muito difícil.

É mais fácil mudar o programa para que desenhe caixas de diferentes tamanhos ou figuras que não se fecham. Faça, por exemplo, com que o valor de **L** na linha 120 varie com o tempo, alterando, assim, o tamanho da caixa; depois, mude o valor de **N** na mesma linha. Atribuindo valores maiores a **N**, a caixa parecerá sólida. Para evitar que as faces da frente e de trás sejam desenhadas, apague a linha 311 (que faz um laço sobre a caixa) e digite:

```
155 GOTO 220
```

Depois, rode o programa.

mas isso complica bastante o programa. Vamos nos limitar, assim, ao desenho isométrico de uma caixa, deixando a perspectiva para um próximo artigo. Como o primeiro e o último conjuntos de coordenadas são bidimensionais, e todas as transformações são lineares (linhas retas ficam retas e linhas paralelas permanecem paralelas), precisaremos alterar apenas as rotinas de inicialização e de desenho.

Caso tenha armazenado em fita ou disco o programa dado no primeiro artigo, carregue-o novamente e digite as linhas que vêm a seguir. É aconselhável gravar todas as listagens, pois, à medida que desenvolvermos o programa em artigos futuros, voltaremos a chamar algumas rotinas. Se você não tem o programa gravado, digite as linhas 5000 a 5130 do artigo anterior desta série. Elas compõem a rotina que desenha uma grade.

Apague as linhas 150 e 155 antes de digitar as que se seguem, mas não rode o programa ainda.



```
9000 LET YX=0: LET XY=0: LET YX
=0: LET YY=1: LET XO=0: LET YO=
0
9010 BORDER 4: PAPER 7: INK 0:
CLS
9070 RETURN
9100 REM MOVER
9110 PLOT XS*XX+XY*YS+XO+127,YX
*XS+YY*YS+YO+76
9120 RETURN
9200 REM DESENHA
9210 DRAW XE*XX+XY*YE+XO+127-PE
EK 23677,YX*XE+YY*YE+YO+76-PEEK
 23678
9220 RETURN
9500 REM LINHA
9510 GOSUB 9100
9520 GOSUB 9200
9550 RETURN
```



```
9000 CLS
9030 XX=1
9040 YY=1
9070 RETURN
9500 LINE(XS*XX+XY*YS+XO+127,YX
*XS+YY*YS+YO+95)-(XE*XX+XY*YE+X
O+127,YX*XE+YY*YE+YO+95),15
9550 RETURN
```

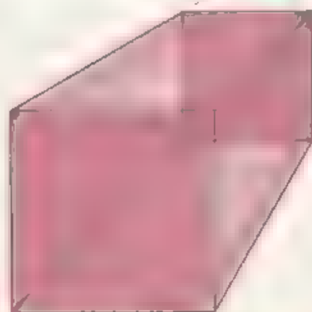


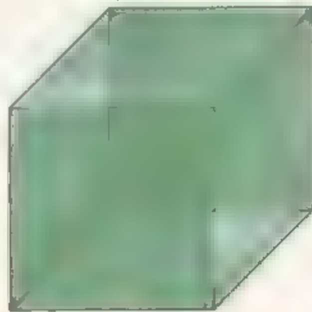
```
9000 XX = 1:XY = 0:YX = 0:YY
1:XO = 0:YO = 0
9010  HGR : HCOLOR= 3
9070  RETURN
9100  REM  MOVE
9108 X1 = XS * XX + XY * YS + X
O + 127
9109 Y1 = YX * XS + YY * YS + Y
O + 76
9110  HPLOT X1,Y1
9120  RETURN
9200  REM  DESENHA
9208 X2 = XE * XX + XY * YE + X
O + 127
9209 Y2 = YX * XE + YY * YE + Y
O + 76
9210  HPLOT X1,Y1 TO X2,Y2
9220  RETURN
9500  REM  LINHA
9510  GOSUB 9100
9520  GOSUB 9200
9550  RETURN
```

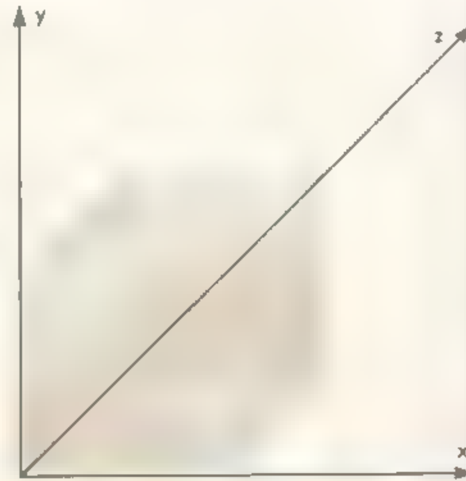


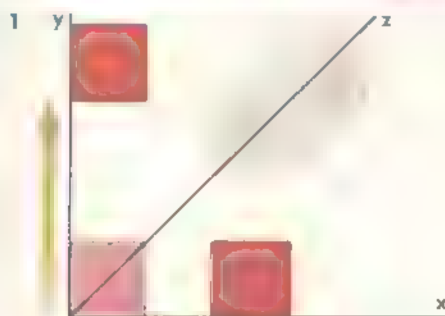
```
9000 PCLS
9030 XX=1
9040 YY=1
9070 RETURN
9500 LINE(XS*XX+XY*YS+XO+127,YX
*XS+YY*YS+YO+95)-(XE*XX+XY*YE+X
O+127,YX*XE+YY*YE+YO+95),PSET
9550 RETURN
```

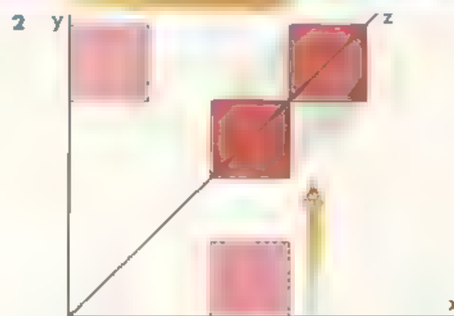
Em perspectiva, os cantos da caixa são convergentes; em projeção isométrica, permanecem paralelos. As duas formas podem ser representadas num sistema de três coordenadas.



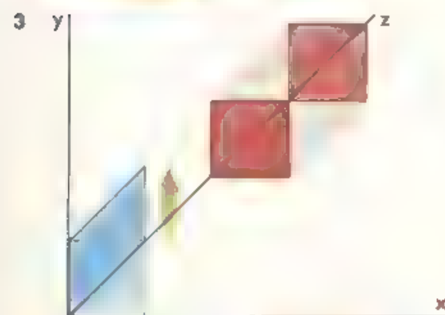


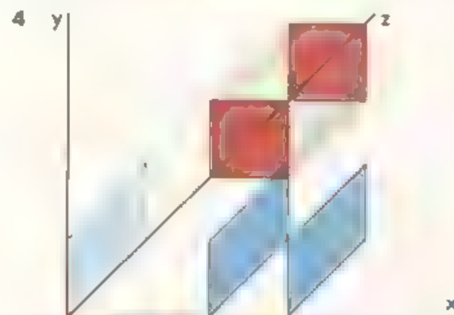


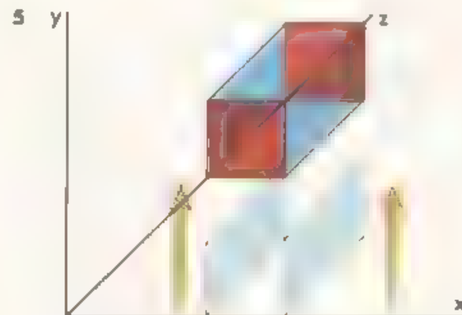


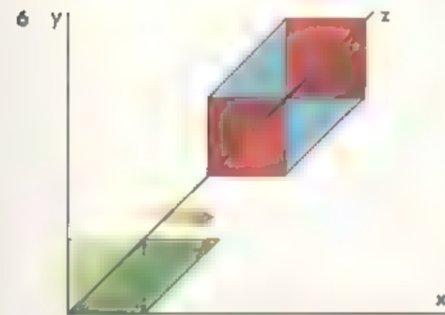


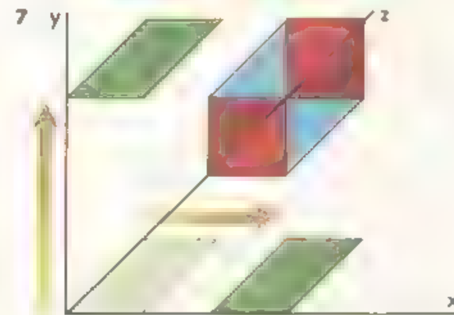
Para representar uma imagem em 3-D num plano bidimensional, um quadrado na origem (1) é deslocado ao longo dos eixos X e Y e, com um novo deslocamento (2), forma as faces da frente e de trás do cubo. Para formar os lados, o quadrado é distorcido (3) na direção Y e depois deslocado (4 e 5) ao longo dos eixos. Da mesma maneira, as faces superior e inferior são obtidas pela distorção (6) na direção X e deslocamento (7 e 8) ao longo dos eixos.

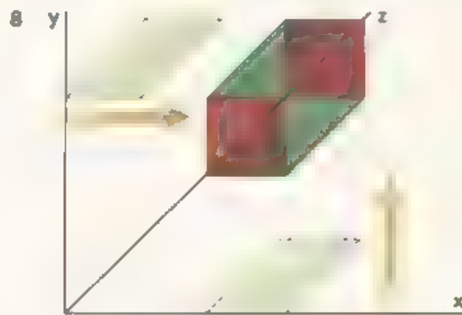












A rotina das linhas 9000 a 9070 inicializa as variáveis de transformação (**XX,XY,YX,XO,YO**) em valores que fazem a rotina de desenho funcionar como antes. No micro TRS-Color não é necessário inicializar variáveis, mas fazê-lo facilita bastante a compreensão do programa.

A rotina movimenta os eixos de maneira que a origem destes esteja no centro da tela. A origem é deslocada porque fica mais fácil supor que o olho do observador está diretamente acima dela no centro da tela. Isso ficará mais claro no próximo artigo, onde veremos como mudar o ponto de vista e acrescentar perspectiva.

Para observar o efeito de cada variável de transformação e ter uma idéia mais clara do que está acontecendo, modifique as linhas 100 a 180:

S

```
100 GOSUB 9000
120 CLS
130 LET XA= 40  LET YA=-20
LET LW 40: LET LH=40: LET NX 5
 LET NY=5
135 GOSUB 5000
140 LET XA=0: LET YA=-20: LET
LW 40  LET LH=40  LET NX=10
LET NY=10
145 GOSUB 5000
160 INPUT "INTRODUZA XX.XY.YX.
YY,XO,YO".XX.XY.YX.YY.XO.YO
170 GOTO 120
180 STOP
```

MSX

```
100 SCREEN 2:COLOR 15,4,4
105 PI=4*ATN(1)
110 GOSUB 9000
120 CLS SCREEN 2
130 XA=-40 YA= 20 LW=40:LH=40:N
X=10 NY=10
135 GOSUB 5000
140 XA=0:YA=-20:LW=40:LH=40:NX=
10 NY 10
145 GOSUB 5000
150 A$=INKEY$:IF A$="" THEN 150
 ELSE SCREEN 0 CLS
160 INPUT"ENTRE XX,XY.YX,YY,XO,
YO ";XX,XY,YX,YY,XO,YO
170 GOTO 120
```

Apple

```
100  GOSUB 9000
120  HOME : HGR   VTAB (24,
130 XA =  - 40:YA =  - 20:LW =
40 LH = 40 NX = 5 NY = 5
135  GOSUB 5000
140 XA = 0:YA =  - 20:LW = 40:L
H = 40:NX = 10:NY = 10
145  GOSUB 5000
160  INPUT "ENTRE XX.XY.YX.YY.X
O,YO ";XX,XY,YX,YY,XO,YO
170  GOTO 120
180  STOP
```

T

```
100 PMODE 4:SCREEN 1,1
105 PI=4*ATN(1)
110 GOSUB 9000
120 CLS:PCLS:SCREEN 1,1
130 XA=-40:YA=-20:LW=40:LH=40:N
X=5:NY=5
135 GOSUB 5000
```

```
140 XA=0:YA=-20:LW=40:LH=40:NX=
10:NY=10
145 GOSUB 5000
150 A$=INKEY$:IF A$="" THEN 150
160 INPUT"INTRODUZA XX,XY,YX,YY
,XO,YO ";XX,XY,YX,YY,XO,YO
170 GOTO 120
```

Ao rodar o programa, você verá um pequeno quadrado no centro da tela, com uma linha pedindo que forneça novos valores para **XX,XY,YX,YY,XO** e **YO** (no MSX, deve-se apertar qualquer tecla para que a linha seja vista). Estas são as variáveis de transformação.

XX e **YY** ajustam os fatores da escala nas direções horizontal e vertical. Se fornecermos valores negativos a essas variáveis, ocorrerá uma reflexão sobre o respectivo eixo. **XO** e **YO** ajustam a posição da grade na tela. **XY** e **YX** são responsáveis por rotações e distorções no desenho. As variáveis de transformação têm valores iniciais **XX**=1, **XY**=0, **YX**=0, **YY**=1, **XO**=0 e **YO**=0. Para modificar a imagem, precisaremos fornecer diferentes valores a essas variáveis; por isso, a linha 160 contém uma declaração **INPUT**.

Aqui estão alguns conjuntos de valores que você pode experimentar no programa. Forneça-os lado a lado, separados por vírgula, exatamente na ordem em que são pedidos, e tecle **<ENTER>** ou **<RETURN>** (no Spectrum, tecle **<ENTER>** após cada um):

–1,0,0,1,0,0: causa uma reflexão em relação ao eixo vertical na área central da tela.

0.5,0,0,2,0,0: aumenta a imagem duas vezes na vertical e diminui pela metade na horizontal.

1,0,0,1,100,–50: move a imagem cem unidades para a direita e cinqüenta unidades para cima (ou para baixo no caso do Spectrum). Usuários do Spectrum podem tentar 1,0,0,1,50,–40 para uma mudança mais significativa.

0,–1,1,0,0,0: faz a imagem rodar em 90 graus.

1,0.5,0,1,0,0: distorce a imagem na horizontal.

1,0.5,0.5,1,0,0: distorce a imagem na horizontal e na vertical.

Para prever o efeito de cada conjunto de valores é preciso algum conhecimento sobre aritmética de matrizes. Portanto, embora saibamos como variar a forma do desenho, é um pouco mais difícil determinar a combinação que produzirá uma forma em 3-D. O próximo programa faz isso com as mesmas rotinas que temos usado. Apague as linhas 135, 145 e 175 do programa anterior antes de inserir esta seção:



```
110 GOSUB 9000
120 LET L=108: LET N=4
130 LET DX=.45*L: LET DY=.3*L
140 LET XN=-(L+DX)/2
150 LET YN=-(L+DY)/2
160 LET XO=XN
170 LET YO=YN
180 GOSUB 500
190 LET XO=XN+DX
200 LET YO=YN+DY
210 GOSUB 500
220 LET XY=DX/L: LET XO=XN
230 LET YY=DY/L: LET YO=YN
240 GOSUB 500
250 LET YO=YN+L
260 GOSUB 500
270 LET XX=DX/L: LET XY=0
280 LET YX=DY/L: LET YY=1: LET
YO=YN
290 GOSUB 500
300 LET XO=XN+L
310 GOSUB 500
320 STOP
500 LET XA=0: LET YA=0: LET LW
=L: LET LH=L: LET NX=N: LET NY
=N
510 GOSUB 5000
520 RETURN
```



```
110 GOSUB 9000
120 L=120:N=4
130 DX=.45*L:DY=.3*L
140 XN=-(L+DX)/2
150 YN=-(L+DY)/2
160 XO=XN
170 YO=YN
180 GOSUB 500
190 XO=XN+DX
200 YO=YN+DY
210 GOSUB 500
220 XY=DX/L:XO=XN
230 YY=DY/L:YO=YN
240 GOSUB 500
250 YO=YN+L
260 GOSUB 500
270 XX=DX/L:XY=0
280 YX=DY/L:YY=1:YO=YN
290 GOSUB 500
300 XO=XN+L
310 GOSUB 500
320 GOTO 320
500 XA=0:YA=0:LW=L:LH=L:NX=N:NY
=N
510 GOSUB 5000
520 RETURN
```



```
100  GOSUB 9000
120 L = 108:N = 4
130 DX = .45 * L:DY = .3 * L
140 XN = - (L + DX) / 2
150 YN = - (L + DY) / 2
160 XO = XN
170 YO = YN
180  GOSUB 500
190 XO = XN + DX
200 YO = YN + DY
210  GOSUB 500
220 XY = DX / L:XO = XN
230 YY = DY / L:YO = YN
240  GOSUB 500
250 YO = YN + L
260  GOSUB 500
270 XX = DX / L:XY = 0
280 YX = DY / L:YY = 1:YO = YN
290  GOSUB 500
300 XO = XN + L
310  GOSUB 500
320  STOP
500 XA = 0:YA = 0:LW = L:LH = L
:NX = N:NY = N
510  GOSUB 5000
520  RETURN
```



```
100 PMODE4:SCREEN 1,1
110 GOSUB 9000
120 L=120:N=5
130 DX=.45*L:DY=.3*L
140 XN=-(L+DX)/2
150 YN=-(L+DY)/2
160 XO=XN
170 YO=YN
180 GOSUB 500
190 XO=XN+DX
200 YO=YN+DY
210 GOSUB 500
220 XY=DX/L:XO=XN
230 YY=DY/L:YO=YN
240 GOSUB 500
250 YO=YN+L
260 GOSUB 500
270 XX=DX/L:XY=0
280 YX=DY/L:YY=1:YO=YN
290 GOSUB 500
300 XO=XN+L
310 GOSUB 500
320 GOTO 320
```

MICRO DICAS

FORA DA TELA

Alguns computadores não conseguem desenhar pontos que estejam fora da tela. Assim, se fornecermos a L um valor muito grande, o programa será interrompido e obteremos uma mensagem de erro. Não é o caso dos micros da linha MSX, os quais desenham qualquer seção de pontos, ignorando aqueles que estiverem fora da tela. Contanto que a imagem esteja inteiramente na tela, os usuários de outros computadores não deverão ter nenhum tipo de problema.

No próximo artigo, veremos como fazer uma checagem no programa, de modo a garantir que somente linhas possíveis sejam desenhadas. Eliminaremos, assim, as interrupções por mensagens de erro.

```
500 XA=0:YA=0:LW=L:LH=L:NX=N:NY
=N
510 GOSUB 5000
520 RETURN
```

Ao rodar o programa, um cubo aparecerá no centro da tela. Seus lados são traçados pela mesma rotina que desenha grades. Esta, com as variáveis adequadas, muda a imagem para a forma desejada. Duas outras variáveis devem ser ajustadas antes que o cubo seja traçado: a variável **L** (que determina qual o tamanho do lado do cubo) e a variável **N** (que determina o número de quadrados na grade).

A variável **DX** especifica o quanto, à direita ou à esquerda, a face de trás está da face da frente; **DY** define o quanto, acima ou abaixo, estas mesmas faces estão uma da outra. **DX** e **DY** fazem com que o cubo pareça ter largura e altura iguais, não adquirindo a forma de um paralelepípedo.

Um outro par de variáveis — **XN** na linha 140 e **YN** na linha 150 — especifica a posição inicial da face da frente do cubo. Esses valores são transferidos para **XO** na linha 160 e **YO** na linha 170. Em seguida, a linha 180 desenha a face de trás. Como ambas as faces são idênticas, para desenhar a face da frente precisamos apenas de um incremento na direção X (linha 190) e outro na direção Y (linha 200).

As linhas 220 e 230 transformam a grade (**XY** e **YX**) e redefinem os valores. Depois, a linha 240 desenha a face de cima. A linha 250 reposiciona verticalmente esse desenho para fazer a face de baixo, que é igual à de cima. De modo similar, as linhas 270 a 310 transformam a grade para obter os lados direito e esquerdo.

Todos os quadrados da grade têm, em cada lado, o mesmo formato e o mesmo tamanho, e todas as linhas paralelas permanecem paralelas, já que não fizemos nenhuma modificação para o desenho em perspectiva, nem adaptamos o programa para mostrar outros ângulos do cubo. Estes serão nossos próximos passos no estudo das técnicas de elaboração de wireframes.

Enquanto não chegamos lá, acrescente um laço ao nosso cubo, transformando-o num pacote de presente. Para isso, mude as linhas 120 e 130 e, depois, adicione as linhas que se seguem. Não se esqueça de gravar o programa!



```
120 LET L=60  LET N=2
130 LET DX= 3*L  LET DY= 4*L
1.. GOSUB 9920
9920 FOR M=0 TO PI STEP  0.
9930 LET D=COS (6*M)
9940 LET S=D*COS M  LET T=D*SIN
 M
9950 LET S=S*20  LET T=ABS  T*2
0
9960 PLOT S+127,T+.06
9970 NEXT M
9980 RETURN
```



```
120 L=60 N=2
130 DX= 6*L DY= .5*L
... GOSUB ...0
10000 REM fita
10010 FOR M=0 TO PI STEP 02
10020 D=COS(6*M)
10030 S=D*COS(M  T=D*SIN(M)
10040 S=S*24 T=ABS T*20
10050 PSET(S+127,T+65),15
10060 NEXT
10070 RETURN
```



```
120 L = 60:N = 2
130 DX = .3 * L:DY = .4 * L
... GOSUB 9920
9920  FOR M = PI TO 2 * PI STEP
 .01
9930 D = COS (6 * M)
9940 S = D * COS (M):T = D *
SIN (M)
9950 S = S * 20:T = - ABS (T
* 20)
9960  HPLOT S + 127,T + 46
9970  NEXT M
9980  RETURN
```



```
100 PMODE4:SCREEN 1,1
110 GOSUB 9000
120 L=60:N=2
130 DX=.6*L:DY=.5*L
140 XN=-(L+DX)/2
150 YN=-(L+DY)/2
160 XO=XN
170 YO=YN
180 GOSUB 500
190 XO=XN+DX
200 YO=YN+DY
210 GOSUB 500
220 XY=DX/L:XO=XN
230 YY=DY/L:YO=YN
240 GOSUB 500
250 YO=YN+L
260 GOSUB 500
270 XX=DX/L:XY=0
280 YX=DY/L:YY=1:YO=YN
290 GOSUB 500
300 XO=XN+L
310 GOSUB 500
311 GOSUB 10000
320 GOTO 320
500 XA=0:YA=0:LW=L:LH=L:NX=N:NY
=N
510 GOSUB 5000
520 RETURN
10000 DRAW"BM151,65"
10010 FOR M=0 TO 4*ATN(1) STEP
.02
10020 D=COS(6*M)
10030 S=D*COS(M):T=D*SIN(M)
10040 S=S*24:T=ABS(T*20)
10050 LINE-(S+127,65-T),PSET
10060 NEXT
10130 RETURN
```



GRÁFICOS: BARRAS E SEGMENTOS

Seja qual for a fonte de seus dados — orçamento doméstico, negócios e até hobbies —, organize-os em gráficos de barras ou de segmentos. Será mais fácil entender o que eles têm a dizer.

Já tratamos, em artigo anterior, da organização de dados em gráficos. Veremos agora como usar gráficos de barras (histogramas) ou de segmentos para mostrar dados numéricos numa forma não linear. Esses dois tipos de gráfico, empregados com freqüência na apresentação de dados estatísticos e comerciais, podem ser coloridos, o que os torna muito mais atraentes.

O gráfico de barras é especialmente indicado para a organização de dados que flutuam bastante. O de segmentos é útil sobretudo quando se quer demonstrar como diferentes valores estão relacionados como partes de um todo. Assim, para fazer a escolha mais adequada, leve em conta o tipo específico de dados de que dispõe.

Os programas que se seguem mostram como empregar o microcomputador na elaboração dos dois tipos de gráfico. Devido à falta de comandos gráficos adequados no ZX-81 e no TRS-80, deixamos de apresentar programas para essas máquinas.

MONTAGEM DO GRÁFICO DE BARRAS

A rotina para montar o gráfico de barras é essencialmente a mesma de qualquer outro gráfico. Uma vez de posse dos dados, é necessário entrá-los no computador, em seguida é preciso definir os eixos, desenhar as barras e acrescentar legendas. Como vimos no artigo da página 481, podemos desenhar as barras à medida que entramos os dados ou entrar todos os dados previamente e só depois traçar o gráfico. Uma terceira opção consiste em ler os dados por meio das instruções **READ/DATA**, alterando-os sempre que se quiser um gráfico diferente. Seja qual for o método escolhido, convém armazenar os dados numa matriz numérica, de modo que o micro possa identificar as coordenadas de cada barra.

LEITURA DOS DADOS

Digite as linhas seguintes para o computador ler os dados. Ao executá-las, não verá nada na tela, mas é bom rodar cada parte do programa para evitar erros. (No Apple ou no TK-2000, para ter de volta a tela de texto, digite **TEXT** e tecle **RETURN**.) De qualquer forma, você pode verificar se aconteceu algo digitando **PRINT A(3)**, por exemplo. Note que os dados já foram armazenados na memória do micro.



```
10 LET n=12
20 DIM a(n)
70 FOR t=1 TO n
80 READ a(t)
90 NEXT t
3010 DATA 3,6,5,9,6,3,6,8,3,5,9
,4
```



```
5 N=12
20 DIM A(N)
60 PMODE 3,1:PCLS:SCREEN 1,0
70 FOR T=1 TO N
80 READ A(T)
90 NEXT
3010 DATA 1,5,3,8,6,2,8,5,2,9,5
,10
```



```
10 N=12
20 DIM A(N)
60 COLOR 10,5,5:SCREEN 2
70 FOR T=1 TO N
80 READ A(T)
90 NEXT
3010 DATA 1,5,6,4,7,9,4,1,5,1,9
,8
```



```
10 N = 12
20  DIM A(N)
60  HOME : HGR
70  FOR T = 1 TO N
80  READ A(T)
90  NEXT
3010  DATA   2,5,3,8,6,2,0,9,7,
8,5,4
```

Neste módulo, o programa seleciona o modo gráfico, determina o número de barras a serem traçadas — 12 (linha 10, ou 5, no TRS-Color) — e dimensiona a matriz para esse número (linha 20).

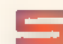


Pode-se usar um número maior para traçar mais barras. Nesse caso, é necessário acrescentar dados à linha 3010, para que haja uma equivalência entre o número de barras e o de dados. É interessante ter um número maior de dados quando se está testando o programa, de modo que se possa aumentar ou diminuir o número de barras sem risco de que ocorra um erro do tipo "falta de dados" ou *end of data*.

- COMO FAZER UM HISTOGRAMA
- GRÁFICO DE BARRAS TRIDIMENSIONAL
- PROGRAME UM GRÁFICO DE SEGMENTOS

A ESCALA DOS EIXOS

O computador conhece agora o valor absoluto de cada barra. Precisamos informá-lo sobre como fazer a escala dos dados, de maneira que cada conjunto deles preencha a tela de forma adequada. Caso contrário, pode-se ter números tão pequenos que praticamente não sejam vistos ou, no outro extremo, valores que caiam fora do intervalo possível da tela. Para fazer a escala, digite as próximas linhas. Novamente nada vai aparecer na tela, mas execute o programa como um teste.



```
  30 LET dx=239/n
  40 READ dy
3000 DATA 18
```



```
30 DX=164/N
40 READ DY
3000 DATA 1
```



```
30 DX 164/N
40 READ DY
3000 DATA 1
```



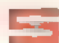
```
30 DX =  INT (170 / N)
40  READ DY
3000  DATA  15
```

Esta parte do programa calcula a escala do eixo X dividindo o comprimento disponível pelo número de barras (linha 30). O valor máximo de barras varia de micro para micro, mas, conhecendo o tamanho de sua tela gráfica e mudando os valores de N, você identificará facilmente o valor mais adequado para um gráfico legível.

Os valores do eixo Y são multiplicados por um fator para que tomem o tamanho mais adequado. Esse valor não

é encontrado automaticamente, mas lido (linha 40) de uma declaração **DATA** (linha 3000). Para cada novo conjunto de valores, avalie qual o melhor fator a ser usado e coloque-o na linha 3000. Nesse ponto, você pode montar uma rotina para dar entrada a tal valor a partir da linha 40 (usando **INPUT**). Se preferir agir assim, não se esqueça de apagar a linha 3000.

O restante do programa é composto de duas rotinas: uma que traça os eixos e outra que desenha as barras. Digite mais estas linhas, mas não as execute, uma vez que as rotinas chamadas ainda não se encontram na memória e um erro será detectado.



```
100 GOSUB 1000
140 GOSUB 2000
160 STOP
```

As linhas 100 e 140 chamam sub-rotinas que traçam os eixos e barras.



```
100 GOSUB 1000
110 FOR T=1 TO N
130 X=(T 1)*DX+18:Y=188 16*A(T)
*DY
140 GOSUB 2000
150 NEXT
160 GOTO 160
```

A linha 100 desvia o programa para a linha que desenha os eixos. A 110 abre um laço que marca o número de barras a serem desenhadas. A cada volta, a linha 130 ajusta a coordenada de **X** — multiplicando-a por **DX** e somando 18 para deixar uma margem à esquerda — e a coordenada de Y — subtraindo do total de linhas disponíveis o valor lido **(A(T))** multiplicado pelo fator **DY** (dado) e pela constante 16. A linha 140 desvia o programa para a rotina que desenha as barras e a 150 fecha o laço, chamando o próximo valor.



```
100  GOSUB 1000
110  FOR T = 1 TO N
130 X = (T - 1) * DX + 8 + A·Y
= 150 - A(T) * DY
140  GOSUB 2000 A   A + 3
150  NEXT
160  END
```

A linha 100 chama a sub-rotina que traça os eixos. A linha 110 abre um laço que determina o número de barras a serem desenhadas. Em seguida, a linha 130 determina a coordenada **X** para o ponto inicial do traçado da barra, multiplicando **T-1** por **DX** e somando **8** — para garantir uma margem à esquerda — e **A** (que provoca um avanço de uma barra em relação à outra — para evitar que as barras fiquem coladas umas às outras). A coordenada **Y** (altura da barra) é calculada subtraindo-se do total de linhas disponíveis o valor lido **(A(T))** pelo fator **DY** (que você determina). A linha 140 chama a sub-rotina que desenha as barras e a seguir incrementa o valor de **A.** A linha 150 fecha o laço, chamando o próximo valor a ser desenhado.

O DESENHO DAS BARRAS

Agora, digite as rotinas que traçam os eixos e desenham as barras:



```
1000 PLOT 16,0: DRAW 0,170
1020 PLOT 16,0  DRAW 235,0
1030 RETURN
2000 FOR a=  TO n
2010 LET s=16+(a 1)*dx
2020 FOR t=s TO s+dx 4
2030 PLOT t,0  DRAW 0,a(a)*dy
2040 NEXT t
2100 NEXT a
2110 RETURN
```

As linhas 1000 e 1020 desenham os eixos, deixando uma margem de dezesseis unidades à esquerda do eixo **Y**. Para desenhar as barras, a linha 2000 toma um valor de cada vez, a linha 2010 coloca o valor na escala **(*dx)** e a linha 2020 monta um laço para desenhar as linhas verticais que formarão a barra. O "-4" no fim da linha provoca um pequeno espaçamento entre as barras. As linhas restantes desenham as linhas que completam a barra e fecham os respectivos laços.



```
1000 LINE(0,25)-(0,191),PSET
1020 COLOR 3
1030 LINE -(255,191),PSET
1050 COLOR 2·FOR T=0 TO 10
1060 LINE(0,191-T*166/10) (3,19
1-T*166/10),PSET
1070 NEXT
1080 RETURN
2000 COLOR 2
2090 LINE(X,190)-(X+DX,Y),PSET,
BF
2100 RETURN
```

A primeira linha desta parte do programa desenha o eixo **Y**, de cima até embaixo, na tela. A linha 1030, por sua vez, se incumbe de traçar o eixo **X**, da esquerda para a direita. Observe que o sinal "—" faz com que seja traçada uma linha da última posição do cursor até a posição indicada entre parênteses. A linha 1050 muda a cor e abre um laço que executa a marcação de uma escala no eixo **Y**.

A rotina seguinte (linhas 2000 a 2100) desenha um retângulo para cada barra e o preenche de amarelo.





```
1000 LINE(8,25) (8,191 ,6
 030 LINF  (255,,91),15
,050 FOR T 0 TO 10
 060 LINF(8,191-T*166/10)-(,0,1
91 T*166/10),15
1070 NEXT
1080 RETURN
2000 COLOR 10,5,5
2090 LINF(X,190) (X+DX,Y),2,BF
2100 RETURN
```

A linha 1000 traça o eixo **Y**, deixando uma margem de oito unidades. A linha 1030 traça o eixo **X**. O laço das linhas 1050 a 1070 marca uma escala no eixo dos **Y** para tornar mais fácil a leitura dos dados.

A rotina das linhas 2000 a 2100 tem como tarefa mudar a cor de frente para amarelo e desenhar um retângulo colorido para cada valor.



```
1000   HCOLOR= 3: HPLOT 5,155 TO
 5,5
1010   HPLOT 5,150 TO 270,150
1020   FOR T = 0 TO 10
1030   HPLOT 2,150 - T * 145 / 1
0 TO 5,150 - T * 145 / 10
1040   NEXT
1050   RETURN
2000   HCOLOR= 5: FOR S = 1 TO D
X
2010   HPLOT X + S,150 TO X + S,
Y
2020   NEXT
2100   RETURN
```

As linhas 1000 e 1010 traçam os eixos **Y** e **X**, respectivamente. As linhas 1020 a 1040 formam um laço para marcar uma escala no eixo **Y**. A rotina seguinte muda a cor para azul e monta um laço que desenhará cada barra como uma sequência de linhas verticais para o valor da barra.

O gráfico de barras é bastante útil na apresentação de dados que flutuam muito num determinado período de tempo — por exemplo, os índices pluviométricos mensais no decorrer de um ano (ao lado). Já o gráfico de segmentos é especialmente indicado quando se quer mostrar como diferentes valores se relacionam enquanto partes de um todo.

TERCEIRA DIMENSÃO

Ao executar esse programa, você verá que a aparência do gráfico de barras é muito menos acadêmica e mais atraente que a dos gráficos lineares, embora os dois tipos sejam construídos com o mesmo número de dados. Mas pode-se melhorar ainda mais a apresentação dos dados com a adição de um efeito tridimensional às barras, o que dará ao desenho um aspecto sólido e natural e um destaque especial às cores.

Antes de entrar o novo programa, grave o que você já tem na memória do computador (para uma posterior comparação). Depois, sem digitar **NEW**, tecle as alterações que se seguem. Se você quiser usar os comandos de edição do seu micro para reduzir o trabalho de digitação, fique atento para não deixar escapar pequenas diferenças que possam existir entre as linhas.



Acrescente ao seu programa apenas algumas linhas. Veja adiante.



```
130 X=(T 1)*D2+18:Y=188 16*A(T)
*DY
1000 LINE (0,191)-(0,25),PSET:L
INE (DX*.6,26 DZ),PSET
1020 PAINT (2,100),4:COLOR 3
1030 LINE -(255-DX*.6,191),PSET
:LINE (255,191 DZ),PSET
1060 LINE (0,191-T*166/10)-(DX*
.6,191-T*166/10-DZ),PSET
2000 COLOR 4
2090 LINE(X,188)-(X+DX,Y),PSET,
BF
```



```
50 DZ 5./(N+ )
115 IF A(T) 0 THEN  60
120 D2 DX*1 4 IF   4(THEND2 DX+
15
130 X (T-,)*DZ+ 8 Y 188  6*A(T
*DY
1000 LINE(8,19,)-(8 25),6 ' NE
 (8+DX* 6,26-DZ),6
,010 LINE  (8+DX* 6,19, DZ ,6 ,
INE  (8,191),6
1020 PAINT (9,180),6
1030 LINE (8, 91  (255 DX* 6, +
1),15 LINE  (255,191 DZ ,15
1040 LINE -(8+DX*.6,191-DZ),15:
LINE  8,191),15 PAINT  ,0 , ,0
),15
1050 FOR T=0 TO 10
1060 LINF(8,,91-T*166/ 0, (8+DX
* 6,191 T*166/10 DZ ,15
```



```
50 DZ = 50 / (N + 1): IF DZ > 5
 THEN DZ = 5
1000  HCOLOR= 3: HPLOT 5,155 TO
 5,5 TO DX * .7,5 - DZ TO DX *
.7,150 - DZ TO 5,150
1010  HPLOT 5,150 TO 270 - DX *
 .7,150 TO 270,150 - DZ TO DX *
 .7,150 - DZ
```

Não execute ainda o programa, pois, como ele não está completo, você obterá apenas uma mensagem de erro. Digite mais estas linhas:



```
1010 DRAW 4,4  DRAW 0,-170
2050 PLOT 16+(a 1,*dx,a(a)*dy
2060 DRAW 4,4
2070 DRAW dx-4,0
2075 DRAW -4,-4: DRAW 4,4
2080 DRAW 0,-a(a)*dy
2090 DRAW -4,-4
```



```
50 DZ=50/(N+1)
115 IF A(T)=0 THEN 160
120 DZ=DX*1 4:IF DX>40 THEN DZ=
DX+15
1010 LINE -(DX*.6,191-DZ),PSET:
LINE-(0,191),PSET
1040 LINE -(DX*.6,191-DZ),PSET:
LINE  (0,191),PSET:PAINT(127,18
9),3
2010 LINE (X+DX,188)-(X+DX*1.6,
188-DZ),PSET:LINE (X+DX*1.6,Y-D
Z),PSET
2020 LINE -(X+DX,Y),PSET:LINE -
(X+DX,188),PSET
2030 PAINT (X+DX+2,185),4
2040 COLOR 3
2050 LINE (X,Y)-(X+DX*.6,Y-DZ),
PSET:LINE-(X+DX*1.6,Y-DZ),PSET
2060 LINE -(X+DX,Y),PSET:LINE -
(X,Y),PSET
2070 PAINT (X+DX/2,Y-1),3
2080 COLOR 2
```



```
2000 COLOR12,5,5
2010 LINE (X+DX,188)-(X+DX*1.6,
188-DZ):LINE -(X+DX*1.6,Y-DZ)
2020 LINE -(X+DX,Y):LINE -(X+DX
,188)
2030 PAINT (X+DX+2,185
2050 LINE (X,Y)-(X+DX*.6,Y-DZ):
LINE -(X+DX*1.6,Y-DZ)
2060 LINE -(X+DX,Y):LINE  (X,Y)
2070 PAINT (X+DX/2,Y 1)
2090 LINE (X,188) (X+DX,Y ,3,BF
```



```
2000  FOR S    1 TO DX
2010  HCOLOR= 3  HPLOT X + S +
DX * .7,Y - DZ TO X + S + DX *
 7,150   DZ: HCOLOR= 6
2020  HPLOT X + S,150 TO X + S,
Y
2030  HPLOT  TO X + S + DX * .7
,Y - DZ: REM  TO X + 1 + DX * .
7,Y - DZ TO X + 1,Y
2040  NEXT
2050  HPLOT  TO X + S + DX * .7
,150 - DZ
```

Você tem agora uma belíssima imagem tridimensional na tela. Como exercício, modifique os valores dos comandos de cores para selecionar aquelas que mais lhe agradam. Faça um registro das linhas que precisam ser mudadas para alterar a escala do gráfico, de modo que você tenha uma referência à mão quando for usar o programa mais tarde. Uma maneira simples de fazer esse registro consiste em utilizar linhas no programa com a declaração **REM**. Ela indica que seu conteúdo é exclusivamente informativo, não devendo, portanto, ser interpretado como instrução ou comando ao computador.

GRÁFICO DE SEGMENTOS

Outra forma interessante de apresentação de dados é o gráfico de segmentos — um tipo de gráfico circular que requer apenas uma coordenada (coordenada polar) para cada informação, representando por meio de um ângulo o tamanho de cada dado.

Estruturalmente, o programa para desenhar um gráfico de segmentos é simples. Para ver como funciona, entre e execute as linhas seguintes.



```
10 DIM a(12)  LET n=0
20 CLS
40 PRINT AT 5,14;"MENU"
50 PRINT AT 8,10;"1- Introduz
ir dados"
60 PRINT AT 10,10;"2-Grafico"
70 PRINT AT 12,10;"3- Saida"
80 LET a$=INKEY$: IF a$<"1"
OR a$>"3" THEN  GOTO 80
90 GOSUB VAL a$*200
100 GOTO 20
200 CLS : LET n=1
210 PRINT "Item No. ";n,:
INPUT  LINE a$: PRINT a$: IF
a$="" THEN  RETURN
220 LET a(n)=VAL a$  LET n=n+1
230 IF n<13 THEN  GOTO 210
240 LET n=n 1: RETURN
400 IF n=0 THEN  RETURN
410 CLS : LET tt=0: FOR t=1 TO
n: LET tt=tt+a(t): NEXT t
420 LET f=(2*PI)/tt
430 CIRCLE 127,86,60
440 LET a=0  FOR k 1 TO n
450 LET m=a+a(k)*f
460 PLOT 127,86: DRAW 60*SIN m
,60*COS m
470 LET a=m
480 NEXT k
490 PAUSE 0: RETURN
```



```
10 DIM A(31),P(31)
15 PMODE 3,1
20 CLS
40 PRINT @45,"MENU"
50 PRINT @169,"1  INTRODUZIR DA
DOS"
60 PRINT @201,"2  GRAFICO"
70 PRINT @233,"3- SAIDA"
80 A$=INKEY$ IF A$<"1" OR A$>"3
" THEN 80
90 ON VAL(A$) GOSUB 200,400,600
100 GOTO 20
200 CLS N=0
210 PRINT"ITEM NO. ",N+1,.INPUT
 A$:IF A$="" THEN RETURN
220 N=N+1:A(N)=VAL(A$)
230 IF N<31 THEN 210
240 RETURN
400 IF N=0 THEN RETURN
410 PCLS:SCREEN 1,0:TT=0:FOR T=
1 TO N:TT=TT+A(T):NEXT
420 FOR T=1 TO N:P(T)=A(T)*810*
ATN(1)/TT:NEXT
430 J=0:P(0)= 1
440 FOR T=1 TO  N
450 IF T=N AND N-3*INT(N/3)=1 T
HEN COLOR 4 ELSE COLOR T-3*INT(
T/3)+2
460 FOR K=1 TO P(T)
470 X=X+.01:Y=Y+ 01
480 LINE (127,95)-(127+60*SIN(X
),95-60*COS(Y)),PSET
490 NEXT K,T
500 IF INKEY$="" THEN 500
510 RETURN
600 CLS
```



```
10 DIM A(31),P(31)
20 CLS
30 LOCATE 18:PRINT"MENU'
40 LOCATE 10,8:PRINT"1   Entrar
 dados"
50 LOCATE 10,11:PRINT"2   Ver g
ráfico"
60 LOCATE 10,14:PRINT"3 - Fim"
80 A$=INKEY$ IF A$<"1" OR A$>"3
" THEN 80
90 ON VAL(A$) GOSUB 200,400,600
100 GOTO 20
200 CLS:N=0
210 PRINT"Item nº ";N+1, INPUT
A(N+1):IF A(N+1)=0 THEN RETURN
220 N=N+1
230 IF N<31 THEN 210
240 RETURN
400 IF N=0 THEN RETURN
410 COLOR 15,15,15 SCREEN 2 TT=
0:FOR T=1 TO N TT=TT+A T) NEXT
420 FOR T=1 TO N P(T)=A(T)*810*
```

```
ATN(1)/TT NEXT
430 J 1 P 0)= 1
440 FOR T=1 TO N
450 IF T NTHEN COLOR 13 ELSE J=
J+1 COLOR J IF J=12 THEN J=0
460 FOR K=. TO P(T)
470 X=X+ 01 Y Y+ 01
480 LINE (127,95)-(127+60*SIN(X
),95 60*COS(Y))
490 NEXT NEXT
500 IF INKEY$=""THEN500
510 RETURN
600 CLS
3000 DATA 8,9,10
```



```
10  DIM A(3 ),P 3 )
20  HOME
40  HTAB 18  PRINT "MENU"
50  HTAB 10  VTAB 10  PRINT "1
- ENTRAR DADOS"
60  HTAB 10  VTAB 1   PRINT "2
- VER GRAFICO"
70  HTAB 10: VTAB 16  PRINT "3
  FIM "
80  GET A$  IF A$  "1" OR A$ >
 "3" THEN 80
90  ON VAL  A$) GOSUB 200,400,
600
100  GOTO 20
200  HOME  N = 0
210  PRINT "ITEM NO  ";N + 1;
INPUT A$  IF A$  "" THEN  RETU
RN
220 N = N + 1 A N)   VAL (A$)
230  IF N   31 THEN 210
240  RETURN
400  IF N = 0 THEN  RETURN
410  HGR :TT = 0: FOR T = 1 TO
N TT   TT + A(T)  NEXT
420  FOR T = 1 TO N P T   A T
 * 810 *  ATN (1   / TT  NEXT
430 J = 0:P(0) =  - 1
440  FOR T = 1 TO N
450 J = J +    IF J   4 THEN J
 .
455  HCOLOR= J: IF T = N THEN
HCOLOR= 5
460  FOR K =   TO P T
470 X = X +  0, Y = Y +  01
480  HPLOT 127,95 TO 127 + 60 *
 SIN  X),95   60 *  COS  Y,
490  NEXT   NEXT
500  GET A$
510  TEXT   RETURN
600  HOME
```

Ao executar o programa, um menu será apresentado na tela, oferecendo três opções: entrar dados, ver o gráfico ou terminar o programa. A linha 80 faz com que o computador espere até que uma tecla seja pressionada. Você pode pressionar qualquer uma, mas, devido à condição imposta pela declaração **IF...THEN** da mesma linha, somente as teclas 1, 2 ou 3 serão aceitas.

Uma condição parecida faz com que o programa volte ao menu, caso você tecle 2 sem ter nenhum dado na memória. A tecla 3, por sua vez, termina o programa. É necessário o comando **RUN** para reiniciá-lo.

No começo da execução, a escolha óbvia é a tecla 1, que lhe possibilita entrar alguns dados. A linha 90 indica para qual ponto do programa a execução será desviada; no caso, ela irá para a linha 200. Cabe a esta linha estabelecer uma variável (**N**) para o número de dados. A entrada de dados se dá na linha 210. Eles permanecerão armazenados numa matriz (**A(.)**), na linha 220. Esta matriz foi dimensionada anteriormente, na linha 10.

Para entrar os dados, digite o número seguido de **ENTER** ou **RETURN**. A linha 230 verifica se o espaço reservado para os dados já não está preenchido. Se ainda há lugar, o programa volta à linha 210 para mais uma entrada. Não é obrigatório ocupar todo o espaço reservado para dados. Quando você quiser encerrar a entrada de números, simplesmente tecle as instruções **ENTER** ou **RETURN** sem ter digitado nenhum número. O programa volta, então, a apresentar o menu principal, por ação da linha 210. Se, por outro lado, você preencheu todo o espaço disponível, a linha 240 encerrará automaticamente a entrada de dados.

CONSTRUÇÃO DO GRÁFICO

Se, agora, você pressionar 2, o programa será desviado para a linha 400, encarregada de iniciar a rotina que desenha o gráfico. A linha 410 lê todos os valores armazenados e calcula o seu total. O restante da rotina calcula uma escala para os dados e desenha os gráficos. Como isso é feito de maneira um pouco diferente para cada computador, a descrição será individual. Para entender melhor o funcionamento das funções circulares no computador, veja o artigo da página 334.



A linha 420 divide o círculo completo (um ângulo de 360 graus ou 2 pi radianos) pelo valor total dos dados para obter um fator de escala. A linha 430 encarrega-se de desenhar o círculo (de raio 60). As linhas 440 e 450 passam por todos os valores, calculando um subtotal para cada um e multiplicando este valor pelo fator de escala (f). O valor encontrado (m) é o valor de pontos na circunferência do círculo. Este é dividido por raios desenhados pela linha 460.

MICRO DICAS

APERFEIÇOE OS PROGRAMAS

Agora que você já completou e testou os programas deste artigo, que tal tentar melhorar a aparência dos gráficos desenhados na tela? Aqui vão algumas sugestões.

Um recurso que enriquece muito o aspecto final de um gráfico — e que aumenta a sua utilidade, sobretudo quando se trata de ilustrar apresentações ou artigos — é a *titulação*. Você pode acrescentá-la ao programa, por meio de linhas **DATA** ou **INPUT**, sob três formas.

- Rótulos para os eixos X e Y: identificam o valor ou nome das divisões dos eixos (exemplo: os nomes dos meses em um gráfico de barras).
- Legendas: indicam o significado de cada barra ou "fatia" do gráfico de segmentos. No gráfico de barras, podem ser colocadas na parte de baixo ou no canto superior direito; no de segmentos, devem aparecer ao lado de cada "fatia".
- Títulos propriamente ditos: especificam o assunto do gráfico — por exemplo, "VENDAS DA COMPANHIA XYZ Resultados de 1985". Costumam ter de uma a três linhas, geralmente centradas e posicionadas na parte superior ou inferior da tela.

Também é interessante salientar uma barra ou segmento, deslocando-o ligeiramente de sua posição — como se uma fatia do gráfico de segmentos fosse retirada. Este recurso, conhecido como *explosão*, pode ser facilmente programado nos micros que possuem o comando **CIRCLE**.



A linha 410 totaliza os dados da memória. Em seguida, a linha 420 passa por todos os dados, calculando um subtotal para cada um e dividindo este subtotal pelo total. Esses valores, já em escala, são armazenados na matriz **P(.)**, que foi dimensionada na linha 10. A linha 430 define variáveis que serão utilizadas na seqüência de cores. O laço **FOR...NEXT** iniciado na linha 440 escolhe a cor para cada um dos segmentos do círculo, e o que começa na linha 460 traça os raios que preencherão os segmentos com a cor predeterminada. A linha 500 faz com que o gráfico fique visível até que alguma tecla seja pressionada, momento em que o menu novamente é apresentado.

GRÁFICOS DA ROM NO SPECTRUM (1)

- O QUE SÃO CARACTERES GRÁFICOS PRÉ-PROGRAMADOS
- COMO ENTRAR CARACTERES GRÁFICOS PELO TECLADO DO MICRO

Os computadores da linha ZX Spectrum dispõem de caracteres gráficos que podem ser usados dentro de um programa ou entrados diretamente pelo teclado. Aprenda a utilizá-los.

Os micros da linha Spectrum (como o TK 90X) são muito versáteis do ponto de vista da programação gráfica. Em artigos anteriores, vimos como a tela gráfica pode ser programada por meio de instruções extremamente poderosas, em alta resolução, tais como **LINE**, **CIRCLE**, **PAINT** etc.

Entretanto, o Spectrum tem um recurso gráfico adicional que é pouco explorado: os *caracteres gráficos* de baixa resolução. Estes podem tanto ser entrados diretamente pelo teclado quanto inseridos por intermédio da função **CHR$**, do BASIC.

O que são caracteres gráficos? Como você já sabe, os caracteres que aparecem no vídeo têm códigos numéricos inteiros, que, teoricamente, podem variar entre 0 e 255. Cada caractere corresponde, portanto, a um byte da memória de vídeo. Parte dessa codificação é convencionada internacionalmente pelo chamado código ASCII, que vai de 32 a 126. Os códigos de 0 a 31 são normalmente utilizados para funções de controle do vídeo, e dependem do tipo de computador. O mesmo ocorre com os códigos de 127 a 255. Nesta faixa, cada fabricante utilizou os códigos de uma maneira, geralmente para acomodar caracteres gráficos.

O Spectrum possui apenas dezesseis caracteres gráficos de teclado, os quais podem ser descritos como combinações dos quatro quadradinhos que formam um caractere (pixels maiores dos que os utilizados na programação gráfica de alta resolução). A matriz básica de um caractere gráfico é:



"Colorindo" individualmente os quadradinhos, em todas as combinações possíveis, obtemos os diferentes gráficos. Por exemplo:



O caractere gráfico com o código 128 é um espaço em branco, o que não o torna menos necessário. Como os demais caracteres, ele pode ser invertido e colorido com **INK** e **PAPER**.

GRÁFICOS DA ROM

Os caracteres gráficos são também conhecidos como *gráficos da ROM*, pois já vêm pré-programados. No Spectrum, eles ocupam a faixa da tabela de caracteres que vai de 128 a 143 e, em geral, são utilizados para a elaboração de desenhos em baixa resolução ou para a composição de bordas e outros tipos de linhas retas. Na formatação de tabelas ou formulários de entrada — seu emprego mais freqüente — apresentam a vantagem de não complicar a programação, misturando tela gráfica com texto. Possibilitam, dessa maneira, o uso de comandos bem mais simples, como **PRINT AT** e **TAB**.

Outra vantagem dos caracteres gráficos pré-programados é que eles se comportam exatamente como qualquer caractere alfanumérico, para fins de impressão na tela. Assim, valem para eles os comandos **INK**, **INVERSE** e **PAPER**, quando se quer mudar as cores de frente e fundo. É divertido tentar compor desenhos de baixa resolução, usando os três comandos acima mencionados e os caracteres gráficos. E, não havendo necessidade de alta resolução, será sempre mais fácil desenhar com eles do que com os comandos gráficos.

ENTRADA PELO TECLADO

Da mesma forma que os caracteres convencionais do ASCII, os caracteres gráficos podem ser digitados pelo teclado. Para isso, pressione simultaneamente as teclas **<CAPS LOCK>** e **<9>** (a tecla 9 é identificada pelo rótulo **GRAPH**, na parte de cima). Desse modo, coloca-se o teclado em modo gráfico, o que é indicado na tela pelo aparecimento do cursor com a letra G. Em seguida, pressione simultaneamente a tecla **<SHIFT>** e uma das teclas numéricas (de 1 a 8). Você obterá, então, um dos símbolos gráficos marcados nas respectivas telas. Se não acionarmos o **<SHIFT>**, aparecerão os símbolos gráficos complementares, ou invertidos, em relação aos marcados nas teclas. Com isso, teremos acesso aos caracteres desejados, que aparecerão diretamente na tela — por exemplo, dentro de uma cadeia alfanumérica.

Quando terminar de digitar os caracteres gráficos em uma linha, pressione novamente **<CAPS LOCK>** e **<9>**, para sair do modo gráfico.

O programa seguinte desenha uma tabela simples, usando blocos gráficos. Depois, coloca os nomes digitados pelo usuário nas linhas da tabela.

```
200 CLS
210 PRINT "▛▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▜"
220 PRINT "▌ NO. ▌    NOME     ▐"
230 PRINT "▙▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▟"
240 FOR i=1 TO 10
250 PRINT "▌     ▌             ▐"
260 NEXT i
270 PRINT "▙▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▟"
280 FOR i=1 TO 10
285 PRINT AT 19,0;"NOME."
290 INPUT n$
300 PRINT AT i+3,4;i
310 PRINT AT i+3,10;n$
320 NEXT i
```

Nas linhas 220 e 250, os espaços em branco e as letras podem ser digitados normalmente, sem ser necessário sair do modo gráfico. Nesse caso, as letras aparecerão apenas em maiúsculo. Para aparecerem em minúsculo, será preciso "desligar" o modo gráfico.

Existem duas desvantagens na entrada de caracteres gráficos pelo teclado: primeiro, nem todos os símbolos disponíveis estão marcados nas teclas, o que torna o processo mais trabalhoso e sujeito a erros; segundo, o programa não pode ser listado em uma impressora não gráfica, ou que não seja própria para o Spectrum.

No próximo artigo desta série, veremos como utilizar caracteres gráficos dentro de programas.

PROGRAMACÃO DE GRÁFICOS EM 3-D (3)

As leis da perspectiva foram [redacted] formuladas no decorrer do Renascimento por artistas como Alberti e [redacted]

[redacted] posição.

Essa flexibilidade é muito útil, sobre- [redacted] "sobrevoando" o objeto.

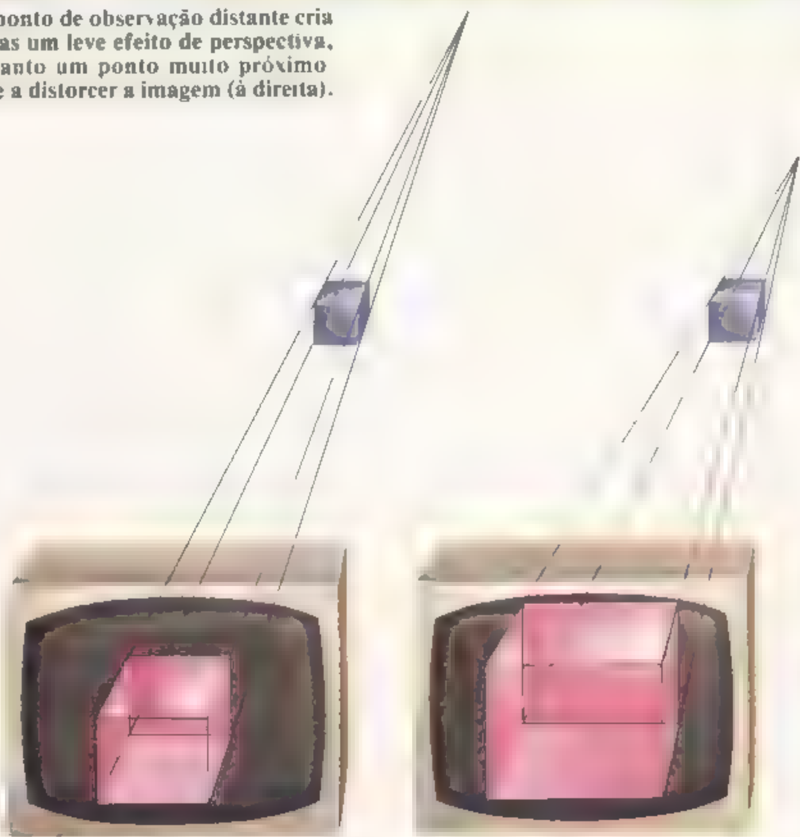
[redacted] uma alta velocidade.

O EFEITO DO PONTO DE OBSERVAÇÃO

Já mostramos como é possível representar um objeto tridimensional em uma tela de duas dimensões, usando convenções visuais que serão interpretadas pelos olhos e pelo cérebro do observador. Os programas anteriores fizeram isso por intermédio de uma projeção isométrica, na qual linhas oblíquas são vistas no contexto do desenho entrando ou saindo da tela.

A forma mais comum de convenção visual é o desenho em perspectiva, em que as linhas que se afastam do observador (ou seja, "entram" na tela) convergem para um ponto distante, conhecido como Ponto de Fuga, e têm diminuídas as suas dimensões, à medida que se afastam do primeiro plano. Na realidade há três pontos de fuga na perspectiva em 3-D: um para cada linha ortogonal desenhada ao longo dos eixos. A posição dos três pontos é fixada com base na relação entre as posições do observador e as do objeto a ser representado (na convenção visual, esses pontos são imaginários).

Assim, a primeira coisa que precisaremos saber é a relação entre as posições do observador e as do objeto. Para determinarmos essa relação, devemos indicar o ponto de vista para os eixos X, Y e Z que foram especificados na tela. Os programas a seguir fazem exatamente isto, executando depois as transformações necessárias.



Um ponto de observação distante cria apenas um leve efeito de perspectiva, enquanto um ponto muito próximo tende a distorcer a imagem (à direita).

ANTES DE COMEÇARMOS

Para rodar esses novos segmentos de programa listados abaixo, recorremos à Rotina da Grade apresentada no artigo *Programação de Gráficos em 3-D (I)* à página 581. Essa rotina deve ser carregada no computador. Uma boa idéia é carregar também a Rotina do Desenho do Círculo que se encontra no mesmo artigo. Embora esta última não seja necessária por enquanto, convém termos todas as rotinas juntas, prontas para o programa global de desenhos que sairá no próximo artigo. Desse modo, as linhas que precisaremos agora são as que vão de 5000 a 6160; todas as outras podem ser apagadas.

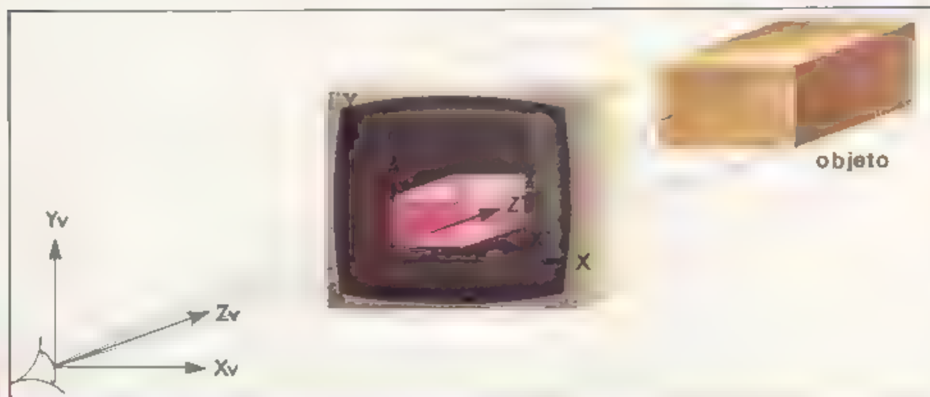
Como anteriormente, teremos que entrar várias porções de programa antes de podermos rodar qualquer coisa, pois a posição dos olhos, da mesma maneira que as transformações para rotação e perspectiva, devem ser implementadas em primeiro lugar.

DEFINIÇÃO DO PONTO DE OBSERVAÇÃO

A primeira parte do programa calcula as variáveis necessárias para se determinar a posição do olho do observador no espaço tridimensional:



```
8000 XV=X:YV=Y:ZV=Z
8010 WV=YV*YV+ZV*ZV
8020 PV=SQR(XV*XV+WV)
8030 IF PV=0 THEN RETURN
8035 WV=SQR(WV)
8040 XU=XV/PV
8050 YU=YV/PV
8060 ZU=ZV/PV
8070 WU=WV/PV
8080 REM ORIENTAÇAO DO OLHO
8090 A=XV*YV:B=ZV:GOSUB 8450·G=
H
8100 A=YV·B=XV:GOSUB 8450:G=G+H
8110 SG=SIN(G)
8120 CG=COS(G)
8130 REM MATRIZ DE ROTAÇAO
8140 R1=WU*CG
8150 R2=-WU*SG
8160 R3=-XU
```

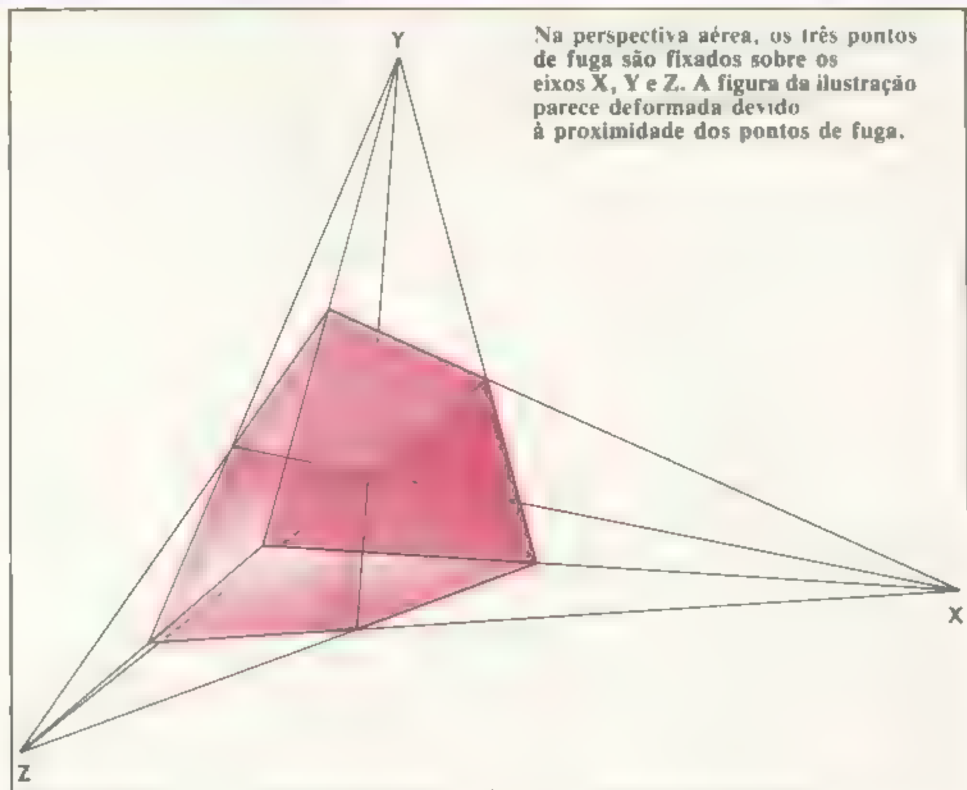


Para representar uma figura em perspectiva, é preciso transformar as coordenadas dos eixos tridimensionais (X1, Y1, Z1) em pontos referidos aos eixos da tela (X, Y).

```
8170 R4=-YU
8180 R5=-ZU
8190 R6=XV*XU+YV*YU+ZV*ZU
8200 IF WU=0 THEN 8340
8210 XT=XV*WU-(YV*YU+ZV*ZU)*XU/
WU
8220 YT=(YV*ZU-ZV*YU)/WU
8230 R7=(ZU*SG-XU*YU*CG)/WU
8240 R8=(-YU*SG-XU*ZU*CG)/WU
8250 R9=CG*XT+SG*YT
8260 S1=(ZU*CG+XU*YU*SG)/WU
8270 S2=(-YU*CG+XU*ZU*SG)/WU
8280 S3=-SG*XT+CG*YT
8330 RETURN
8340 REM CASO ESPECIAL NO EIXO
X
8350 R7=-1
8360 R8=0
8370 R9=0
8380 S1=0
8390 S2=1
8400 S3=0
8410 RETURN
8450 IF B<>0 THEN H=ATN(A/B):RE
TURN
8460 H=PI/2:RETURN
```



```
8000 XV = X:YV = Y:ZV = Z
8010 WV = YV * YV + ZV * ZV
8020 PV =  SQR (XV * XV + WV)
8030  IF PV = 0 THEN  RETURN
8035 WV =  SQR (WV)
8040 XU = XV / PV
8050 YU = YV / PV
8060 ZU = ZV / PV
8070 WU = WV / PV
8080  REM  ORIENTACAO DO OLHO
8090 A = XV * YV:B = ZV: GOSUB
8450:G = H
8100 A = YV:B = XV: GOSUB 8450:
G = G + H
8110 SG =  SIN (G)
8120 CG =  COS (G)
8130  REM  MATRIZ ROTACAO
8140 R1 = WU * CG
8150 R2 =  - WU * SG
8160 R3 =  - XU
8170 R4 =  - YU
8180 R5 =  - ZU
8190 R6 = XV * XU + YV * YU + Z
V * ZU
8200  IF WU = 0 THEN 8340
8210 XT = XV * WU - (YV * YU +
ZV * ZU) * XU / WU
8220 YT = (YV * ZU - ZV * YU) /
 WU
8230 R7 = (ZU * SG - XU * YU *
CG) / WU
8240 R8 = ( - YU * SG - XU * ZU
 * CG) / WU
8250 R9 = CG * XT + SG * YT
8260 S1 = (ZU * CG + XU * YU *
SG) / WU
8270 S2 = ( - YU * CG + XU * ZU
 * SG) / WU
8280 S3 =  - SG * XT + CG * YT
8330  RETURN
8340  REM  CASO ESPECIAL NO EIX
O-X
8350 R7 =  - 1
8360 R8 = 0
```



Na perspectiva aérea, os três pontos de fuga são fixados sobre os eixos X, Y e Z. A figura da ilustração parece deformada devido à proximidade dos pontos de fuga.

```
8370 R9 = 0
8380 S1 = 0
8390 S2 = 1
8400 S3 = 0
8410  RETURN
8450  IF B <  > 0 THEN H =  ATN
 (A / B): RETURN
8460 H = PI / 2: RETURN
```



```
8000 LET XV=X· LET YV=Y· LET ZV
=Z
8010 LET WV=YV*YV+ZV*ZV
8020 LET PV SQR (XV*XV+WV)
8030 IF PV=0 THEN  RETURN
8035 LET WV=SQR WV
8040 LET XU=XV/PV
8050 LET YU=YV/PV
8060 LET ZU=ZV/PV
8070 LET WU=WV/PV
8080 REM ORIENTACAO
8090 LET A=XV*YV: LET B=ZV: GOS
UB 8450: LET G=H
8100 LET A=YV  LET B=XV· GOSUB
8450: LET G=G+H
8110 LET SG=SIN G
8120 LET CG=COS G
8140 LET R1=WU*CG
8150 LET R2= WU*SG
8160 LET R3=-XU
8170 LET R4=-YU
8180 LET R5=-ZU
8190 LET R6=XV*XU+YV*YU+ZV*ZU
8200 IF WU=0 THEN  GOTO 8340
8210 LET XT=XV*WU (YV*YU+ZV*ZU)
*XU/WU
8220 LET YT=(YV*ZU-ZV*YU)/WU
8230 LET R7=(ZU*SG-XU*YU*CG)/WU
8240 LET R8=(-YU*SG-XU*ZU*CG)/W
U
8250 LET R9=CG*XT+SG*YT
8260 LET S1 (ZU*CG+XU*YU*SG)/WU
8270 LET S2=(-YU*CG+XU*ZU*SG)/W
U
8280 LET S3=-SG*XT+CG*YT
8330 RETURN
8350 LET R7=-1
8360 LET R8=0
8370 LET R9 0
8380 LET S1=0
8390 LET S2=1
8400 LET S3=0
8410 RETURN
8450 IF B<>0 THEN  LET H=ATN (A
/B)  RETURN
8460 LET H=PI/2  RETURN
```



```
8000 XV=X:YV=Y:ZV=Z
8010 WV=YV*YV+ZV*ZV
8020 PV=SQR(XV*XV+WV)
8030 IF PV=0 THEN RETURN
8035 WV=SQR(WV)
8040 XU=XV/PV
8050 YU=YV/PV
8060 ZU=ZV/PV
8070 WU=WV/PV
8080 REM ORIENTACAO
8090 A=XV*YV:B=ZV:GOSUB 8450:G=
H
8100 A=YV:B=XV:GOSUB 8450:G=G+H
8110 SG=SIN(G)
8120 CG=COS(G)
8130 REM ROTACAO DA MATRIZ
8140 R1=WU*CG
8150 R2=-WU*SG
```

```
8160 R3=-XU
8170 R4=-YU
8180 R5=-ZU
8190 R6=XV*XU+YV*YU+ZV*ZU
8200 IF WU=0 THEN 8340
8210 XT=XV*WU-(YV*YU+ZV*ZU)*XU/
WU
8220 YT=(YV*ZU-ZV*YU)/WU
8230 R7=(ZU*SG-XU*YU*CG)/WU
8240 R8=(-YU*SG-XU*ZU*CG)/WU
8250 R9=CG*XT+SG*YT
8260 S1=(ZU*CG+XU*YU*SG)/WU
8270 S2=(-YU*CG+XU*ZU*SG)/WU
8280 S3=-SG*XT+CG*YT
8330 RETURN
8340 REM CASO ESPECIAL NO EIXO
X
8350 R7=-1
8360 R8=0
8370 R9=0
8380 S1=0
8390 S2=1
8400 S3=0
8410 RETURN
8450 IF B<>0 THEN H=ATN(A/B) EL
SE H=PI/2
8460 RETURN
```

Para entendermos o que está acontecendo, devemos nos lembrar que o eixo Z é aquele que sai do centro da tela e vem na direção do observador; o eixo Y aponta para cima da tela e o eixo X, para os lados.

A posição dos olhos está em (**XV, YV, ZV**). A variável **WV** nos dá a distância nas direções Y e Z combinadas. Definimos a origem em (**0, 0, 0**) para a posição do objeto que observamos no espaço. Para simplificar, situamos essa posição no centro da tela. A linha 8030 interromperá a rotina, caso a posição dos olhos seja definida na origem. As variáveis **XU**, **YU**, **ZU** são as distâncias, na direção dos respectivos eixos (X, Y, X) de uma linha de comprimento unitário entre a origem e a posição dos olhos. **WU** é a distância nas direções Y e Z combinadas.

Se movermos o objeto da esquerda para a direita, nosso ângulo de visão sofrerá alterações. Esse ângulo é medido a partir do eixo X e definido nas linhas 8090 e 8100. Um teste é feito nas linhas 8450 e 8460 para evitar divisões por zero, que interromperiam o programa, gerando uma mensagem de erro.

Quando olhamos diretamente ao longo do eixo X, temos uma visão comum do objeto, mas, se rodarmos nosso ponto de observação, o objeto também rodará, proporcionando ângulos mais interessantes. As linhas 8140 a 8280 inicializam as variáveis que definem a orientação da posição do olho no espaço. Essa posição é tal que o seu eixo Z encontra-se ao longo da linha que vai do olho até a origem da tela.

O resto da rotina inicializa variáveis para casos especiais; por exemplo, quando a posição do olho estiver exatamente sobre o eixo X.

AS TRANSFORMAÇÕES

A próxima seção transforma as coordenadas X, Y e Z do cubo nas coordenadas finais da tela. Essa transformação leva em conta a posição do olho e o efeito da perspectiva.



```
8500 X1=T1*X+T4*Y+T7
8510 Y1=T2*X+T5*Y+T8
8520 Z1=T3*X+T6*Y+T9
8530 REM 2-D PARA 3-D
8540 X2=R1*X1+R7*Y1+R8*Z1+R9
8550 Y2=R2*X1+S1*Y1+S2*Z1+S3
8560 Z2=R3*X1+R4*Y1+R5*Z1+R6
8570 REM OBJETO PARA OLHO
8575 IF Z2<ZN THEN RETURN
8580 X3=D*X2/Z2
8590 Y3=D*Y2/Z2
8600 RETURN
```



```
8500 X1 = T1 * X + T4 * Y + T7
8510 Y1 = T2 * X + T5 * Y + T8
8520 Z1 = T3 * X + T6 * Y + T9
8530  REM  2-D PARA 3-D
8540 X2 = R1 * X1 + R7 * Y1 + R
8 * Z1 + R9
8550 Y2 = R2 * X1 + S1 * Y1 + S
2 * Z1 + S3
8560 Z2 = R3 * X1 + R4 * Y1 + R
5 * Z1 + R6
8575  IF Z2 < ZN THEN  RETURN
8580 X3 = D * X2 / Z2
8590 Y3 =  - D * Y2 / Z2
8600  RETURN
```



```
8500 LET X1=T1*X+T4*Y+T7
8510 LET Y1=T2*X+T5*Y+T8
8520 LET Z1=T3*X+T6*Y+T9
8540 LET X2=R1*X1+R7*Y1+R8*Z1+R
9
8550 LET Y2=R2*X1+S1*Y1+S2*Z1+S
3
8560 LET Z2=R3*X1+R4*Y1+R5*Z1+R
6
8575 IF Z2<ZN THEN  RETURN
8580 LET X3 D*X2/Z2
8590 LET Y3= D*Y2/Z2
8600 RETURN
```

Essa rotina usa uma matriz aritmética complicada para transformar as coordenadas (X, Y), que são basicamente os passos de transformação descritos no artigo anterior desta série — *Programação de Gráficos em 3-D (2)*. As linhas 8500 a 8520 transformam o plano cartesiano (X, Y) de duas dimensões (2-D) no espaço de três dimensões (3-D) (X1, Y1, Z1). As linhas 8540 a 8560 transformam as coodernadas espaciais (X1, Y1, Z1), de modo a posicioná-las de acordo com a localização do olho e a direção (X2, Y2, Z2).



A linha 8575 testa se a nova posição a ser encontrada está excessivamente próxima do olho (ou seja, se Z2 está entre 0 e ZN) ou se está situada atrás dos olhos. Em ambos os casos, a posição é ignorada, pois seria impossível visualizarmos o objeto. Se a posição estiver muito distante de ZN (a partir da posição em frente ao olho), então a localização das coodernadas da tela (X3, Y3) será calculada somente no fim da rotina. O parâmetro **D/Z2** nessas linhas acrescenta perspectiva à figura, projetando o objeto em uma tela plana a uma distância definida **D**. Se estabelecermos um valor pequeno para **D**, o efeito obtido será uma perspectiva mais nítida

(como se a tela estivesse mais próxima do usuário); se **D** for grande, a tela parecerá estar a uma grande distância do observador, de modo que o efeito de perspectiva será pequeno e a imagem aparecerá distorcida, mesmo quando vista obliquamente, de qualquer direção

COMO INICIALIZAR AS VARIÁVEIS

O proximo passo será reescrever as Rotinas de Inicialização e Desenho para podermos usar as novas rotinas de transformação.

MSX

```
9000 CLS
9020 XM=256·YM=192
9030 XD=XM/2:YD=YM/2
9040 ZN=1
9042 CLS INPUT"Distância do pla
no de projeção: ";D
9045 IF D<0 THEN D=1000*ZN
9050 T1=1:T2=0·T3=0
9060 T4=0:T5=1:T6=0
9070 T7=0·T8=0:T9=0
9085 CLS
9090 RETURN
9500 X=XS·Y=YS:GOSUB 8500:IF Z2
<ZN THEN 9520
9510 IX=128+X3·IY=97-Y3
9520 X=XE:Y=YE·GOSUB 8500:IF Z2
<ZN THEN 9550
9530 LINE(127+X3,95-Y3)-(IX,IY)
,15
9550 RETURN
```



```
9000  HOME
9020 XM = 280:YM = 192
9030 XD = XM / 2:YD = YM / 2
9040 ZN = 1
9042  INPUT "DISTANCIA AO PLANO
 DE PROJECAO ";D
9045  IF D < 0 THEN D = 1000 *
ZN
9050 T1 = 1:T2 = 0:T3 = 0
9060 T4 = 0:T5 = 1:T6 = 0
9070 T7 = 0:T8 = 0:T9 = 0
9090  HOME : RETURN
9500 X = XS:Y = YS: GOSUB 8500:
 IF Z2 < ZN THEN 9520
9505  IF X3 <   139 OR Y3 <  -
 95 OR X3 > 140 OR Y3 > 96 THEN
 9550
9510 IX = 139 + X3:IY = 95 + Y3

9520 X = XE:Y = YE: GOSUB 8500:
 IF Z2 < ZN THEN 9550
9525  IF X3 <  - 139 OR Y3 <  -
 95 OR X3 > 140 OR Y3 > 96 THEN
 9550
9530  HPLOT 139 + X3.95 + Y3 TO
```

```
 IX,IY
9550  RETURN
```



```
9000 CLS
9020 LET XM=256: LET YM=176
9030 LET XD=XM/2: LET YD=YM/2
9040 LET ZN=1
9042 INPUT "IMTRODUZA DISTANCIA
 AO PLANO DE PROJECAO",D
9045 IF D<=0 THEN  LET D=1000*Z
N
9050 LET T1=1: LET T2=0: LET T3
=0
9060 LET T4=0: LET T5=1: LET T6
=0
9070 LET T7=0: LET T8=0: LET T9=
 0
9090 CLS : RETURN
9500 LET X=XS: LET Y=YS: GOSUB
8500: IF Z2<ZN THEN  GOTO 9520
9505 IF X3<-127 OR Y3<-87 OR X3
>128 OR Y3>88 THEN  GOTO 9550
9510 PLOT 127+X3,87+Y3
9520 LET X=XE: LET Y=YE: GOSUB
8500: IF Z2<ZN THEN  GOTO 9550
9525 IF X3<-127 OR Y3<-87 OR X3
>128 OR Y3>88 THEN  GOTO 9550
9530 DRAW 127+X3-PEEK 23677,87+
Y3-PEEK 23678
9550 RETURN
```



```
9000 PCLS
9020 XM=256:YM=192
9030 XD=XM/2:YD=YM/2
9040 ZN=1
9042 CLS:INPUT " INTRODUZA A DI
STANCIA AO PLANO  DE PROJECAO "
;D
9045 IF D<=0 THEN D=1000*ZN
9050 T1=1:T2=0:T3=0
9060 T4=0:T5=1:T6=0
9070 T7=0:T8=0:T9=0
9085 CLS
9090 RETURN
9500 X=XS:Y=YS:GOSUB 8500:IF Z2
<ZN THEN 9520
9505 IF X3<-127 OR Y3<-96 OR X3
>128 OR Y3>95 THEN 9550
9510 DRAW"BM"+STR$(INT(127+X3))
+","+STR$(INT(95-Y3))
9520 X=XE:Y=YE:GOSUB 8500:IF Z2
<ZN THEN  9550
9525 IF X3<-127 OR Y3<-96 OR X3
>128 OR Y3>95 THEN 9550
9530 LINE -(127+X3,95-Y3),PSET
9550 RETURN
```

A linha 9020 atribui às variáveis **XM** e **YM** as dimensões máximas da tela nas direções X e Y, respectivamente. A linha 9030 calcula o ponto médio; a 9040 calcula a posição mais próxima permitida nos pontos a serem desenhados, conforme a posição do olho.

A variável **D** nos dá a distância atual da posição do olho até o plano de projeção (a tela) e determina a perspectiva. Os valores de **D** são inseridos quando rodamos o programa, de modo que podemos variar o grau da perspectiva. Se nenhum valor for especificado e pressionarmos **ENTER** ou **RETURN** a linha 9045 adotará o valor 1000. As linhas 9050 a 9070 atribuem valores para as constantes de transformação, especificando o plano no qual será desenhada a imagem em perspectiva.

Esse trecho do programa continua com a Rotina de Desenho revisada. Na linha 9500, são inicializadas as variáveis para as constantes de transformação; nas linhas 9505 e 9525 é feito um teste para determinar se o novo ponto a ser desenhado (linhas 9510 e 9530) está dentro dos limites da tela. O MSX não precisa desse teste, pois ele ignora os pontos fora da tela. Nos outros micros, o teste serve para prevenir contra mensagens de erro.

CHAME A ROTINA

Precisaremos agora da rotina para desenhar a grade fornecida no primeiro artigo da série. Caso você não a tenha gravado, digite-a antes de inserir esta parte final do programa.



```
100 PI=4*ATN(1)
110 GOSUB 9000
120 L=20:N=5
125 GOSUB 505:GOTO 140
130 GOSUB 500
140 IF X=0 AND Y=0 AND Z=0 THEN
 170
150 GOSUB 1000
160 GOTO 130
170 CLS
180 END
500 IF INKEY$="" THEN 500
505 CLS
510 INPUT"Posição do observador
 (X,Y,Z): ";X,Y,Z
520 GOSUB 8000
530 CLS:SCREEN 2
540 RETURN
1000 P=L/2
1010 T1=1:T2=0:T3=0
1020 T4=0:T5=1:T6=0
1030 T7=-P:T8=-P:T9=-P
1040 GOSUB 1200 'BAIXO
1050 T7=-P:T8=-P:T9=P
1060 GOSUB 1200 'CIMA
1070 T4=0:T5=0:T6=-1
1080 GOSUB 1200 'ESQUERDA
1090 T7=-P:T8=P:T9=P
1100 GOSUB 1200 'DIREITA
1110 T1=0:T2=-1:T3=0
1120 GOSUB 1200 'TRAS
1130 T7=P:T8=P:T9=P
1140 GOSUB 1200 'FRENTE
1170 RETURN
1200 XA=0:YA=0:LW=L:LH=L:NX=N:N
Y=N
1210 GOSUB 5000
1220 RETURN
```



```
100 PI = 4 *  ATN (1)
110  GOSUB 9000
120 L = 20:N = 5
125  GOSUB 505: GOTO 140
130  GOSUB 500
140  IF X = 0 AND Y = 0 AND Z =
 0 THEN 170
150  GOSUB 1000
160  GOTO 130
170  HOME
180  TEXT : END
500  GET T$: IF T$ = "" THEN 50
0
505  TEXT : HOME
510  INPUT "POSICAO DO OBSERVAD
OR (X,Y,Z) ";X,Y,Z
520  GOSUB 8000
530  HGR2 : HCOLOR= 3
540  RETURN
1000 P = L / 2
1010 T1 = 1:T2 = 0:T3 = 0
1020 T4 = 0:T5 = 1:T6 = 0
1030 T7 =  - P:T8 =  - P:T9 =
- P
1040  GOSUB 1200: REM  BAIXO
1050 T7 =  - P:T8 =  - P:T9 = P

1060  GOSUB 1200: REM  CIMA
1070 T4 = 0:T5 = 0:T6 =  - 1
1080  GOSUB 1200: REM  ESQUERDA

1090 T7 =  - P:T8 = P:T9 = P
1100  GOSUB 1200: REM  DIREITA
1110 T1 = 0:T2 =  - 1:T3 = 0
1120  GOSUB 1200: REM  TRAS
1130 T7 = P:T8 = P:T9 = P
1140  GOSUB 1200: REM  FRENTE
1170  RETURN
1200 XA = 0:YA = 0:LW = L:LH =
L:NX = N:NY = N
1210  GOSUB 5000
1220  RETURN
```



```
110 GOSUB 9000
120 LET L=20: LET N=3
125 GOSUB 505: GOTO 140
130 GOSUB 500
140 IF X=0 AND Y=0 AND Z=0
THEN  GOTO 170
150 GOSUB 1000
160 GOTO 130
170 CLS
180 STOP
500 IF INKEY$=""  THEN   GOTO
 500
505 CLS
510 INPUT "INTRODUZA POSICAO D
O OBSERVADOR (X,Y,Z)",X,Y,Z
520 GOSUB 8000
530 RETURN
1000 LET P=L/2
1010 LET T1=1: LET T2=0: LET T3
=0
1020 LET T4=0: LET T5=1: LET T6
=0
1030 LET T7=-P: LET T8=-P: LET
T9=-P
```

```
1040 GOSUB 1200  REM BAIXO
1050 LET T7= P  LET T8= P: LET
T9=P
1060 GOSUB 1200: REM CIMA
1070 LET T4=0: LET T5=0: LET T6
= 1
1080 GOSUB 1200: REM ESQUERDA
1090 LET T7=-P: LET T8=P: LET T
9=P
1100 GOSUB 1200  REM DIREITA
1110 LET T1=0  LET T2= 1: LET T
3=0
1120 GOSUB 1200  REM TRAS
1130 LET T7=P  LET T8=P: LET T9
=P
1140 GOSUB 1200: REM FRENTE
1170 RETURN
1200 LET XA=0: LET YA=0: LET LW
 L  LET LH=L  LET NX=N  LET NY=
N
1210 GOSUB 5000
1220 RETURN
```



```
100 PI=4*ATN(1):PMODE 4,1
110 GOSUB 9000
120 L=20:N=5
125 GOSUB 505:GOTO 140
130 GOSUB 500
140 IF X=0 AND Y=0 AND Z=0 THEN
 170
150 GOSUB 1000
160 GOTO 130
170 CLS
180 END
500 IF INKEY$="" THEN 500
505 CLS
510 INPUT" INTRODUZA A POSICAO
DO OBSERVA- DOR (X,Y,Z) ";X,Y,Z
520 GOSUB 8000
530 PCLS:SCREEN 1,1
540 RETURN
1000 P=L/2
1010 T1=1:T2=0:T3=0
1020 T4=0:T5=1:T6=0
1030 T7=-P:T8=-P:T9=-P
1040 GOSUB 1200 'BAIXO
1050 T7=-P:T8=-P:T9=P
1060 GOSUB 1200 'CIMA
1070 T4=0:T5=0:T6=-1
1080 GOSUB 1200 'ESQUERDA
1090 T7=-P:T8=P:T9=P
1100 GOSUB 1200 'DIREITA
1110 T1=0:T2=-1:T3=0
1120 GOSUB 1200 'ATRAS
1130 T7=P:T8=P:T9=P
1140 GOSUB 1200 'FRENTE
1170 RETURN
1200 XA=0:YA=0:LW=L:LH=L:NX=N:N
Y=N
1210 GOSUB 5000
1220 RETURN
```

Agora, rode o programa. Se ele estiver funcionando corretamente, o computador acessará a rotina de inicialização que começa na linha 110. Essa rotina inicializa algumas variáveis e imprime uma mensagem que pede um valor para **D**, a distância do plano de projeção. A perspectiva nesse programa é feita de tal maneira que, quanto maior a distância, mais afastado aparecerá o objeto. Devemos entrar com o valor 1000 para começar. O programa retorna para a linha 120, que especifica o comprimento (**L**) de cada lado do cubo, e o número de grades quadradas (**N**) ao longo de cada lado. As linhas 130 a 160 lêem as coordenadas da posição do olho (linha 510), e desenham o cubo a partir dessa posição. Assim que obtivermos a primeira imagem do objeto desenhado, podemos digitar um novo conjunto de coordenadas e ver o cubo por um outro ângulo. Quando entramos os valores 0, 0 e 0, o programa termina. A rotina das linhas 500 a 530 chama a rotina de posição (linha 120) para inicializar as constantes de transformação. A rotina do Cubo (linhas 1000 a 1170) posiciona e desenha cada um dos seis lados. A rotina Lado (linhas 1200 a 1220) desenha cada lado como uma grade, usando a Rotina da Grade.

Tente valores diferentes para a posição do olho e estude os efeitos decorrentes. O valor 1000 para **D** e 250, 0 e 0 para X, Y e Z são bons para começar. Depois, adote 100 para **D** e 25,0 e 0 para X, Y e Z. O cubo deverá aparecer do mesmo tamanho, mas a perspectiva será muito mais pronunciada. Para digitar novos valores para X, Y e Z devemos pressionar qualquer tecla e a mensagem aparecerá. Para atribuir novos valores a **D**, interrompa o programa com <**BREAK**> ou <**RESET**> e rode-o novamente. Uma mensagem aparecerá para **D**, seguida de outra para X, Y e Z.

Como o programa ignora eventuais erros ocasionados por pontos fora da tela, os resultados podem ser bastante estranhos (neste caso, uma linha inteira é omitida, mesmo quando apenas uma de suas extremidades excede o ponto máximo permitido). Para pontos mais próximos do que uma certa distância mínima e pontos localizados atrás do olho, ZN será ignorado. Dessa maneira, se a posição do olho estiver muito próxima, ou mesmo dentro do cubo, poderão ocorrer falsos resultados.

Finalmente, é aconselhável guardar uma cópia da listagem completa em fita ou disco, de modo a usar as mesmas rotinas para desenhar formas duplicadas e produzir gráficos circulares em futuros programas.

CONECTE UMA IMPRESSORA

Ao contrário das máquinas de escrever, sempre prontas para serem operadas, as impressoras só podem ser usadas com um computador depois que certas condições são satisfeitas.

Se você leu *Como Escolher uma Impressora* (página 521), já sabe como agir para adquirir um desses periféricos.

Dependendo da marca e do modelo do micro, além da impressora, você precisará provavelmente de uma interface que a compatibilize com ele.

Mas adquirir uma impressora e sua respectiva interface pode não resolver todos os problemas. Com efeito, muitos usuários descobrem, após a compra, que ainda precisam de um bom tempo para ligar o novo periférico e fazê-lo funcionar de modo fluente. É que isso exige informações complementares sobre quais são os códigos de controle adequados, como enviá-los à impressora e como trabalhar com todas aquelas funções especiais, que pareciam prometer tanto e que o fizeram sair de casa e adquirir a impressora.

Neste artigo, tentaremos esclarecer alguns dos pontos ainda obscuros e orientá-lo no sentido de resolver as dificuldades mais comuns que surgem para aqueles que desejam se equipar com uma dessas máquinas.

COMO AJUSTAR A IMPRESSORA

Suponhamos que você já adquiriu uma impressora, que a conectou à tomada, e que está agora no ponto em que não sabe como fazer para colocar tudo aquilo em funcionamento...

O primeiro conselho é: não ligue ainda o periférico. Existem uma ou duas coisas que você deve fazer primeiro; para muitas impressoras, o mais importante de tudo é a remoção do parafuso de fixação para transporte (usado para imobilizar a máquina quando ela é levada de um lugar para outro).

Algumas impressoras não têm mecanismo de proteção para transporte; portanto, verifique com bastante atenção o manual de instruções do fabricante. Geralmente, existem dois ou três parafusos, que podem ser auxiliados por elásticos ou fitas adesivas. Estes devem ser removidos cuidadosamente, pois as últimas coisas de que você vai necessitar quando estiver imprimindo serão elásticos e fitas adesivas. Quanto aos parafusos, mantenha-os à mão: você talvez precise reutilizá-los em caso de transporte da impressora.



- COMO AJUSTAR A IMPRESSORA
- QUE PAPEL USAR
- TESTE A IMPRESSORA
- A CONEXÃO COM O COMPUTADOR
- COMO FAZER O CONTROLE
- AS OPÇÕES DE HARDWARE
- EFEITOS ESPECIAIS
- CÓDIGOS DE ESCAPE
- LIGUE O ZX 81 E O SPECTRUM
- IMPRESSORAS DO FUTURO

Uma vez destravado o aparelho, você pode ligá-lo. O botão para isso geralmente está em uma das laterais ou na parte traseira da máquina.

Muitas impressoras dispõem de uma pequena luz, vermelha ou verde, destinada a avisar se a fonte está ligada. Mas, ainda que ela não seja provida de luzes, você provavelmente perceberá o pequeno movimento que a cabeça de impressão realiza para ajustar-se. Certos modelos avisam quando o papel não está colocado e ajustado corretamente, por meio da emissão de um alarma sonoro ou outro sinal qualquer.

Se a impressora não funcionar na primeira tentativa, não se desespere: pode ser alguma coisa tão simples como um fio solto. Se, ao contrário, ela funcionar (o que é, aliás, muito mais provável), desligue-a novamente e coloque o papel (e a fita impressora, caso isto seja necessário).

O PAPEL

Há dois tipos básicos de impressora quanto ao tipo de papel usado: as que empregam formulários contínuos (papel dobrado em z, com perfurações nas margens) e as que utilizam folhas soltas ou papel alimentado por bobinas. As máquinas mais versáteis — como as compatíveis com a linha Epson — aceitam os dois tipos de papel. Impressoras térmicas e eletrostáticas requerem papel especial e em geral são máquinas específicas para determinadas marcas de computador.

Se você puder escolher, não use papel muito fino ou excessivamente grosso. E, quando for ajustá-lo na impressora, verifique primeiro se ele não está frouxo ou demasiado tenso, pois o ajuste incorreto pode provocar rompimentos no papel ou levá-lo a enredar-se no mecanismo.

AUTOTESTE

No caso de sua impressora dispor de recursos para um autoteste, é conveniente acioná-los assim que ela estiver pronta para trabalhar. Esse tipo de verificação consiste em imprimir ininterruptamente um conjunto completo de caracteres. Ele serve para dois propósitos: primeiro, mostrar que a impressora está funcionando independentemente do computador; segundo, para exibir o conjunto de caracteres disponíveis no periférico.

Nas impressoras de tipo mais comum, o autoteste é acionado quando se pressiona o botão de avanço de linha ou de formulário.

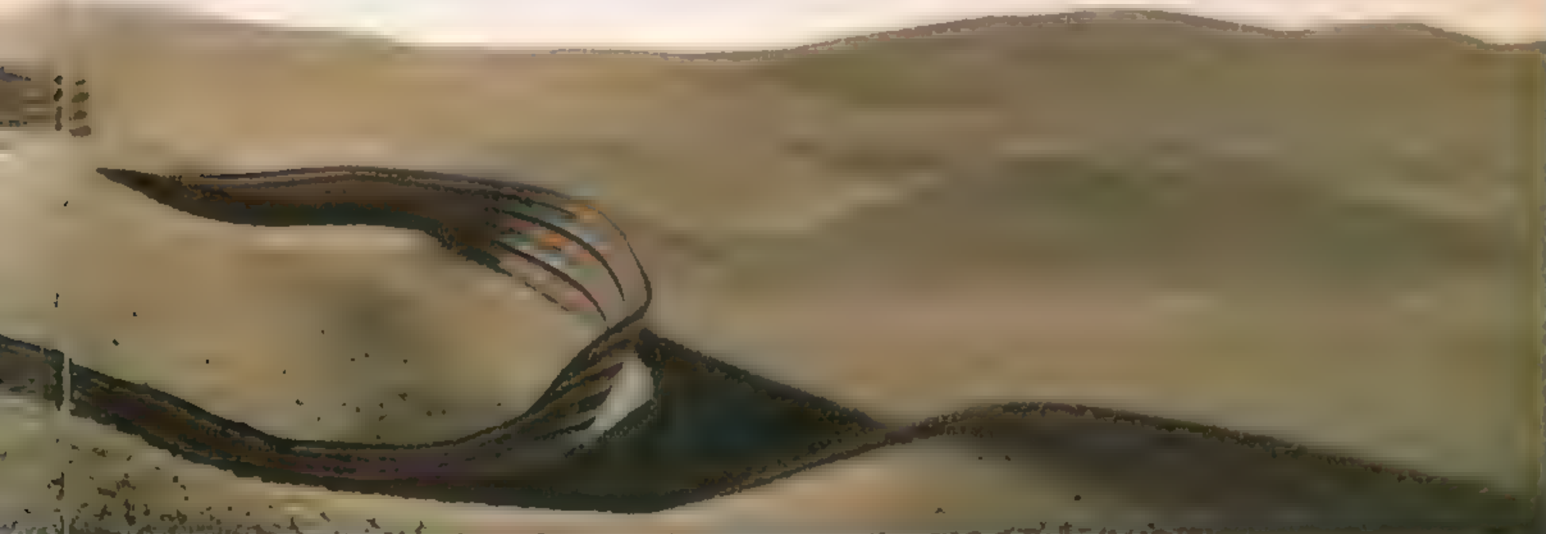
Para parar o autoteste, basta desligar a impressora. (Algumas máquinas dispõem de um botão especial para acionar o autoteste.)

FITAS DE IMPRESSÃO

A menos que a sua impressora seja térmica ou eletrostática, ela provavelmente precisará de uma fita de impressão. Essas fitas costumam ser vendidas em cartuchos descartáveis, e não é difícil trocá-las (consulte antes o diagrama existente no manual da impressora). Um pequeno botão na parte de cima do cartucho da fita serve para dar a ela a tensão adequada.

A CONEXÃO AO COMPUTADOR

A última e mais importante etapa dessa sequência é a ligação da impressora ao computador. Em muitos casos isso é tão simples quanto inserir o cabo de conexão no soquete correto: essa é a forma de ligação das impressoras que usam o padrão Centronics (paralelo). Freqüentemente, como acontece com o TK-2000 e com o MSX, existe um conector claramente reservado para a impressora. Nos computadores da linha Sinclair (ZX-81 e Spectrum), é necessário uma interface especial, que é conectada no mesmo ponto destinado a ligar outros tipos de expansão (memória, sintetizadores de voz etc.). Nesse caso, é preciso, quase sempre, desligar uma expansão já existente, para poder colocar a interface da impressora, a não ser que



você tenha comprado uma interface com um conector adicional de expansão. Esse sistema é conhecido como "encadeamento em margarida" e deve ser feito com certa precaução.

Se você vai adquirir uma impressora que não é específica para o seu computador, deve tomar alguns cuidados. Se o seu micro é um MSX, ou um TK-2000, por exemplo, a impressora deve ser, obrigatoriamente, do tipo paralelo, padrão Centronics (a maioria das impressoras mais populares, como a Mônica, a Grafix e outras compatíveis com os modelos Epson e Seikosha tem esse padrão). Os micros da linha Apple podem trabalhar tanto com impressoras paralelas como com as seriais, que usam o padrão RS-232C, dependendo da interface que se vai utilizar. Já os micros compatíveis com o TRS-Color usam apenas impressoras seriais.

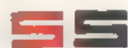
Ao conectar o cabo, você notará que a maioria das impressoras tem soquetes que impedem uma ligação em posição invertida. Isso não ocorre com a impressora para os micros da linha TRS-80 (por exemplo, o CP-500), que aceitam o soquete em qualquer posição.

Se você quiser conectar uma impressora de fabricante independente nos micros da linha Sinclair (Spectrum e ZX-81), precisará adquirir uma interface especial. As interfaces para tais micros existentes no mercado nacional devem ser ligadas a impressoras paralelas com o padrão Centronics.

Em alguns casos, a interface só funciona quando o computador é carregado com um programa específico para isso. Entretanto, uma vez que você tenha uma, poderá ligar praticamente qualquer impressora ao seu computador. As melhores interfaces são oferecidas pelos próprios fabricantes de micros, mas mesmo assim é bom fazê-las funcionar antes de adquiri-las.

COMANDOS DE CONTROLE

Qualquer coisa a ser impressa tem que ser enviada primeiro pelo computador à impressora, através do cabo de conexão. Os comandos usados para controlar a impressora variam de computador para computador, e em geral consistem de algumas instruções simples.



O Spectrum e o ZX-81 se comunicam com a impressora por comandos simples em BASIC, de fácil manipulação. O comando **LPRINT** tem um efeito similar ao **PRINT**; com ele, porém, a saída vai para a impressora e não para a tela. No ZX-81, todas as regras do **PRINT** valem também para o **LPRINT**. No Spectrum, entretanto, não se tem a mesma liberdade de especificar o tipo de impressão, como se faz com o comando **PRINT**. Por exemplo, não é possível utilizar os comandos **PAPER**, **INK**, **OVER**, **FLASH** e **BRIGHT**. O **INVERSE** só funciona em certos tipos de impressoras gráficas. O mesmo acontece com os caracteres gráficos que podem ser colocados dentro de um **PRINT**, tanto no ZX-81 quanto no Spectrum (as impressoras não específicas para tais micros não dispõem desses caracteres no seu conjunto de impressão). O **PRINT AT** também não funciona na impressora.

Assim como o comando **PRINT** é substituído por **LPRINT**, o **LIST** dá lugar a **LLIST**, que lista o programa da memória na impressora. Quando a listagem ocupar mais de uma tela, **LLIST** gerará uma impressão contínua, independentemente do tamanho do programa.

O terceiro comando BASIC usado para acionar a impressora é o **COPY**. Ele copia o que estiver na tela por um processo denominado *descarregamento de tela*. Quando a impressora é utilizada para produzir pontos em vez de letras é chamada de impressora gráfica. Você pode trabalhar com o **COPY** para fazer uma cópia da tela contendo os resultados de um texto ou de uma listagem de programa em qualquer tipo de impressora. A cópia de telas gráficas funcionará apenas se a impressora for específica para o seu computador (uma impressora gráfica, portanto).

O TK-90X emprega uma interface paralela, própria para acionar uma impressora e que exige o carregamento prévio de um programa de controle *(driver)*. Esse programa é vendido em conjunto com a interface, com um cabo de conexão e com um manual de instruções. O software só precisa ser carregado uma vez ao se ligar o computador. Mas, sempre que este for desligado, será preciso carregá-lo novamente. O software permite a utilização dos comandos **LPRINT**, **LLIST** e **COPY**.

Já os micros da linha ZX-81, como o TK-85, requerem também uma interface paralela; mas não é preciso carregar um software de controle para usar diretamente os comandos citados.



Os micros da linha Apple podem ser conectados tanto a impressoras seriais como a impressoras paralelas. Para isso, é necessário adquirir separadamente a interface adequada de controle; esta deve ser encaixada em um dos soquetes internos do console do computador. De forma geral, costuma-se dar preferência à interface paralela, uma vez que é maior o número de impressoras que seguem esse padrão.

A impressora é tratada, pelo BASIC e pelo sistema operacional, como um dispositivo de saída que recebe o número lógico correspondente ao soquete no qual foi inserida a placa de controle. Assim, se encaixarmos a placa no soquete 1 (a localização mais comum) e quisermos ativar a impressora, é necessário digitar o comando em BASIC:

```
PR#1
```

A partir desse momento, todo **PRINT** ou **LIST** que for digitado em modo direto, ou encontrado dentro de um programa, terá sua saída direcionada tanto para a tela quanto para a impressora (modo eco). Portanto, o **PR** pode ser colocado em pontos selecionados de um programa, para "ligar" e "desligar" o eco na impressora.

Os micros da linha Apple não dispõem, como os de outras linhas, de comandos específicos para a impressora, como **LLIST**, **LPRINT** etc. Por motivos óbvios, os comandos de posicionamento do cursor, como **HTAB** e **VTAB** não funcionam com a impressora.



A interface de comunicação adequada aos micros da linha MSX é do tipo paralela padrão Centronics. Tanto ela quanto o conector para o cabo de ligação já estão incluídos na configuração básica da máquina.

Existem três protocolos de impressão (conjuntos de caracteres de impressão diferentes) para micros da linha MSX. O primeiro padrão é o MSX Internacional, que segue a tabela de caracteres ASCII especiais e gráficos do padrão MSX original. Esse padrão é seguido por mais de trinta fabricantes de todo o mundo. O segundo é o ABICOMP (Associação Brasileira de Indústrias de Computadores): ele segue normas nacionais, que incluem os caracteres acentuados da língua portuguesa. Finalmente, temos um terceiro protocolo, criado no Brasil pela Gradiente para os primeiros modelos do Expert MSX. Ao se adquirir uma impressora, é necessário especificar qual o padrão seguido pelo micro; caso contrário, a impressora não funcionará corretamente.

Os comandos usados com o BASIC

para o controle da impressora são também bastante simples e diretos.

O comando **LPRINT** tem um efeito similar ao **PRINT**; com ele, porém, a saída vai para a impressora e não para a tela. Todas as regras de sintaxe que valem para o **PRINT** servem para o **LPRINT**, inclusive quando se trata do emprego de **TAB** e **USING**. Entretanto, os comandos de especificação de cores (como o **COLOR**) e os de localização bidimensional do cursor (como o **LOCATE**) não funcionam na impressora. O mesmo acontece com os caracteres gráficos e especiais que podem ser colocados dentro de um **PRINT**, uma vez que impressoras não específicas para esse micro não dispõem desses caracteres no seu conjunto de impressão.

Assim como o comando **PRINT** é substituído por **LPRINT**, o comando **LIST** é substituído por **LLIST**, que lista o programa da memória na impressora. Quando a listagem tomar mais de uma tela, **LLIST** gerará uma impressão contínua independentemente do tamanho do programa. As regras de utilização do **LIST** valem para o **LLIST**. Outra função útil, de uso exclusivo da impressora, é a **LPOS**, que informa ao programa a posição lógica da cabeça de impressão, em número de colunas a partir da margem esquerda de uma linha.



Também nos micros da linha TRS-80 a interface de comunicação é do tipo paralela padrão Centronics, e ela e o conector para o cabo de ligação incluem-se na configuração básica das máquinas mais completas, como o CP-500.

Os comandos usados com o BASIC para o controle da impressora são simples e diretos. O **LPRINT** tem um efeito similar ao **PRINT**, embora direcione a saída para a impressora e não para a tela. Todas as regras de sintaxe do **PRINT** valem para o **LPRINT** (o que inclui as expressões **TAB** e **USING**). Entretanto, os comandos de localização bidimensional do cursor como **PRINT@** não funcionam com a impressora. O mesmo se dá com os caracteres gráficos e especiais que podem ser colocados dentro de um **PRINT**, já que impressoras não especificas para o TRS-80 não têm esses sinais no seu conjunto de impressão.

Assim como o comando **PRINT** dá lugar a **LPRINT**, o **LIST** é substituído por **LLIST**, que lista o programa da memória na impressora. Se a listagem tomar mais de uma tela, **LLIST** gerará uma impressão contínua independentemente do tamanho do programa. As regras do **LIST** valem para o **LLIST**.

Outra função útil, de uso exclusivo da impressora, é a **LPOS**, que informa o programa a respeito da posição lógica da cabeça de impressão, em número de colunas a partir da margem esquerda.



Os micros da linha TRS-Color dispõem só de uma porta serial para comunicação com periféricos, que é a usada para conexão com a impressora. Mesmo assim, o controle da impressão (feita por alguns comandos do BASIC) nesses micros também é simples. Para listar um programa digite **LLIST** (seguido por <*ENTER*>). Todas as regras do **LIST** valem para o **LLIST**, só que a saída vai para a impressora, e não para a tela. Para usar **PRINT** com a impressora, é preciso digitar a expressão:

```
PRINT#-2,"TEXTO"
```

Ela determina ao computador que envie sua saída (aqui a palavra **TEXTO**) ao periférico **-2**, que é o número de identificação para a impressora (mais exatamente, para a interface serial). Repare que há uma vírgula após o **-2** no comando acima. Ela informa ao computador que a seguir vem um texto para ser impresso. Além disso, se usado com impressoras capazes de imprimir letras minúsculas, o texto que está em vídeo inverso na tela aparecerá em minúsculas na impressora (é necessário pressionar **<SHIFT><0>** antes).

Assim, o comando **PRINT#-2** tem um efeito similar ao **PRINT** (com uma diferença: ele determina que a saída vá para a impressora, e não para a tela). As regras de sintaxe do **PRINT** valem para o **PRINT#-2**, inclusive quando aparecem as expressões **TAB** e **USING**. Porém, os comandos de especificação de cores, e de localização bidimensional do cursor, como **PRINT@**, não funcionam com a impressora. O mesmo se dá com os caracteres gráficos e especiais que podem ser colocados dentro de um

PRINT, pois as impressoras não específicas para o TRS-Color não têm esses sinais em seu conjunto de impressão.

Outra função de uso exclusivo da impressora é a **POST(-2)**: informa a posição lógica da cabeça de impressão, em número de colunas a partir da margem esquerda de uma linha.

PROGRAMAÇÃO INTERNA DA IMPRESSORA

Quem deseja dominar os segredos do trabalho com impressoras precisa conhecer primeiro como se faz sua programação interna e externa. A programação interna é feita normalmente por meio de microinterruptores (os *DIP*) e fios removíveis de conexão (os *estrapes*), situados na placa interna da máquina. Já a externa é realizada através de sequências de caracteres de controle (denominados *seqüências de escape* por começarem com o código ASCII 27, correspondendo ao caractere de controle chamado ESCAPE).

Dentro das impressoras e de outros equipamentos eletrônicos, você encontrará um conjunto de pequenas chaves: as chaves DIP. Elas podem ser usadas para alterar o estado da impressora, quando ela for ligada (o estado *default*, ou predefinido). São comumente inicializados por chave o comprimento da linha, o espacejamento, o retorno do carro e o comprimento da página. As impressoras de tecnologia japonesa permitem ainda que você selecione o conjunto de caracteres internacionais de que precisa (exemplo: o sinal da libra esterlina, ou acentos para certos caracteres, empregados em idiomas como francês, sueco e português).

Para encontrar os interruptores de programação interna, consulte o manual da impressora: em geral, eles estão localizados sob a tampa da impressora (esta, porém, só deve ser removida quando a impressora estiver desconectada da corrente elétrica).

Uma vez localizadas as chaves DIP, você pode trocá-las de posição empurrando-as delicadamente com os dedos ou com uma chave de fenda. O manual de instruções deve ter um diagrama do DIP e de suas posições, de modo a mostrar para que elas estão selecionadas. Faça um teste você mesmo.

Outras características de uma impressora (como, por exemplo, o fato de todo caractere de alimentação de linha ser seguido automaticamente de um retorno de carro) podem ser alteradas removendo-se ou inserindo-se um estrape (pequeno fio em forma de U, com conectores em ambas as pontas) em pontos apropriados da placa eletrônica da máquina. Antes de fazer isso, porém, é conveniente consultar o manual específico da sua impressora.

CÓDIGOS DE ESCAPE

Algumas impressoras (como a Mônica, por exemplo) admitem o envio de códigos de controle, da mesma forma que se enviam os dados a serem impressos. Esses códigos são freqüentemente chamados de sequências de escape. Assim, muitas impressoras esperam que você coloque o código para o escape, **CHR$ (27)**, antes do código de controle. Isso permite à impressora saber que o que está chegando no momento é uma mensagem de programação externa, e não um caractere a ser impresso.

Cada impressora tem as suas sequências de escape próprias, embora os caracteres usados nas máquinas compatíveis com a Epson estejam se tornando rapidamente um padrão para impressoras que oferecem facilidades similares. Nas máquinas mais versáteis há um enorme conjunto de seqüências de escape. Estas podem ser enviadas por um comando **LPRINT** normal.

Em uma Epson, o exemplo a seguir imprime a palavra COMPUTADOR em modo enfatizado, desde que tenham sido enviados os comandos corretos.



```
LPRINT CHR$(27);'E';'COMPUTADOR'
```



```
PR#1
PRINT CHR$(27);'E';'COMPUTADOR'
```



```
PRINT#-2,CHR$(27);'E';'COMPUTADOR'
```

Mas não se esqueça de voltar ao modo normal. Para isso, use os seguintes comandos:



```
LPRINT CHR$(27);'F'
```



```
PR#1
PRINT CHR$(27);'F'
```



```
PRINT#-2,CHR$(27);'F'
```

Lembre-se, porém, de que esses códigos são exclusivamente para as impressoras Epson e as que são compatíveis com ela. Para outras impressoras, os códigos serão diferentes.

OS CÓDIGOS EPSON

Os efeitos mais comuns obtidos com os códigos de escape do tipo Epson são os seguintes: impressão em itálico, impressão grifada, impressão em negrito, subscritos e sobrescritos. Certas operações variam de máquina para máquina. As que apresentamos a seguir funcionam na maioria das impressoras compatíveis com a Epson:

ESC 4 ativa um conjunto alternativo de caracteres (geralmente o itálico).
ESC 5 retorna ao conjunto padrão.
ESC - ativa e desativa o grifo. Deve ser seguido por **CHR$(1)** ou por **CHR$(49)**; para a alteração, é desativado por **CHR$(0)** ou **CHR$(48)**, conforme tenha sido ativado por **CHR$(1)** ou por **CHR$(49)**
ESC G ativa o modo dupla impressão.
ESC H desativa o modo dupla impressão.
ESC S, quando é seguido de **CHR$(0)**, ativa o modo de sobrescrição; quando é seguido de **CHR$(1)**, ativa o modo de subscrição.
ESC T retorna a impressora ao modo normal depois de ativado o modo de subscrição ou sobrescrição.
ESC @ inicia a impressora e volta todos comandos aos valores originais.

Você pode usar as sequências de escape em praticamente todas as impressoras conhecidas, colocando-as depois do comando que o seu computador usa para enviar uma saída à impressora (**LPRINT, PRINT #-2**, ou qualquer outro) e separando-as por pontos e vírgulas, assim como se faria com qualquer outro comando **CHR$**.

Algumas impressoras exigem que se coloque um caractere **ASCII** para o ESCAPE entre todos os códigos de controle. O manual de instruções deve dizer se você vai ou não precisar disso.

NOVAS TENDÊNCIAS

As sequências de escape terão um papel mais importante na medida em que as impressoras se tornarem mais baratas e sofisticadas. Já existem impressoras com matrizes de ponto cuja qualidade é quase a mesma das impressoras "margarida"; algumas delas têm características próprias usualmente associadas com fotocomposição profissional, tais como: espacejamento proporcional e justificação automática à direita ou à esquerda.

FELIZ ATERRISSAGEM

- UMA ROTINA DE CONTROLE DO AVIÃO PELO TECLADO
- A ATERRISSAGEM
- ACIDENTES AÉREOS
- PERDAS E DANOS

Agora que você já é um (relativo) ás da aviação, desligue o piloto automático e, com apenas três funções, faça o aparelho aterrissar. Mas tenha cuidado com os urubus.

Última parte de nosso programa simulador de vôo, este artigo fará com que você possa finalmente assumir o controle direto do avião. Três pares de teclas permitirão aumentar e diminuir a rotação do motor, virar o aparelho para os lados, subir ou descer.

Nossa viagem começou a 20.000 metros do centro da pista e a 2.000 metros de altura.

Pilotar pode ser uma tarefa difícil... sobretudo se você nada sabe do assunto. Não espere bons resultados nos primeiros vôos: como em tudo na vida, a habilidade só vem com o tempo.

Relativamente lentos por estarem em BASIC, programas como esse parecem ainda mais morosos pelo fato de não permitirem mudanças bruscas de curso. Com paciência e perseverança, porém, você conseguirá controlar o aparelho e melhorar seus pousos. O programa o manterá informado a respeito de cada aterrissagem ou acidente. Essas mensagens são importantes: será por meio delas que você saberá não só o que está acontecendo em cada momento como também onde se encontram suas principais deficiências.



```
3000 LET POW=0: LET K$=INKEY$:
IF K$="" THEN RETURN
3010 IF K$="S" THEN LET POW=-1
3020 IF K$="F" THEN LET POW=1
3030 IF K$="Q" THEN LET PT=PT+
1
3040 IF K$="A" THEN LET PT=PT-
1
3050 IF K$="O" AND RI>-30 THEN
 LET RI=RI-1
3060 IF K$="P" AND RI<30 THEN
LET RI=RI+1
```

```
3070 RETURN
5020 CLS : INPUT "VELOCIDADE DO
 VENTO (1-50 M/S)",X0
5025 IF X0>50 OR X0<1 THEN GOT
O 5020
5030 INPUT "DIRECAO DO VENTO (0
-359 GRAUS)",X1
5035 IF X1>359 OR X1<0 THEN GO
TO 5030
5040 LET X0=X0/3
5050 PRINT ''VELOCIDADE DO VENT
O= ",3*X0;" M/S": PRINT "DIRECA
O= ",X1;" GRAUS"
5055 PAUSE 100: CLS
5060 LET WY= X0*COS (X1*C)
5070 LET WX=-X0*SIN (X1*C)
5500 GOSUB 3000: GOSUB 1000
5510 IF PZ<=0 THEN GOTO 6000
5520 GOSUB 2000
5530 GOTO 5500
6000 IF ABS RL>RT OR PT>TP OR P
T<0 OR VV>80 THEN GOTO 6030
6010 IF ABS PX>WR OR ABS PY>100
0 THEN GOTO 6060
6020 CLS : PRINT " PARABENS!":
PRINT "ATERRISSAGEM BEM SUCEDID
A ": GOTO 6100
6030 FOR N=0 TO 20 STEP .5: PLO
T 127,130: DRAW 120-INT (RND*24
0),45-INT (RND*9): SOUND .005,
20-N: NEXT N
6040 PAUSE 50
6050 CLS : PRINT "LAMENTAMOS IN
FORMAR QUE O VOO  PJ 26 SOFREU
 UM GRAVE ACIDENTE  NAO HA SOBR
EVIVENTES ": GOTO 6100
6060 CLS : PRINT "VOCE POUSOU F
ORA DA PISTA"
6070 IF VV<40 THEN PRINT "COMO
 SUA VELOCIDADE ERA BAIXA  QUA
SE NADA ACONTECEU ": GOTO 6100
6080 IF VV<80 THEN PRINT "ENTR
E MORTOS E FERIDOS         SAL
VARAM SE TODOS ": GOTO 6100
6090 PRINT "NINGUEM SOBREVIVE A
 UM POUSO   NESSA VELOCIDADE!"
6100 PRINT '"VALORES RELATIVOS
AO POUSO"
6110 PRINT '"VELOCIDADE DO AR
",INT VV," M/S": PRINT "DISTANC
IA= ",INT (SQR (PY*PY+PX*PX)):
PRINT "INCLINACAO FRONTAL= ";PT
6120 PRINT "INCLINACAO LATERAL=
 ",RL: PRINT "ROTACAO DO MOTOR=
 ",INT (10*TC)/10," X 1000"
6130 PRINT "DESVIO DO EIXO DA P
ISTA= ",INT ABS PX," METROS": P
RINT "DIRECAO  ",AD;" GRAUS"
6140 PRINT '"QUER VOAR NOVAMENT
E (S/N) ?"
6150 LET A$=INKEY$: IF A$<>"S"
AND A$<>"N" THEN GOTO 6150
6160 IF A$="N" THEN CLS : STOP
6170 RUN
```

A sub-rotina que vai de 3000 a 3070 nos dará o controle do aparelho.

Usando o comando **INKEY$** para detectar teclas, podemos modificar a rotação do motor, virar o aparelho, subir ou descer. Uma nova variável — **POW** — aparece nas linhas 3010 e 3020; ela serve para alterar a rotação. Um valor zero é atribuído a **POW** na linha 3000 cada vez que a sub-rotina é chamada. A tecla S determina a velocidade, enquanto F acelera o motor.

Q e A são usadas para provocar a ascensão e a descida do avião, somando ou subtraindo 1 à variável **PT** nas linhas 3030 e 3040. Finalmente, podemos usar O ou P para virar o aparelho — as linhas 3050 e 3060 cuidam disto, alterando o valor de **RL**.

As linhas 5020 a 5070 permitem variar o nível de dificuldade do jogo, escolhendo a força e a direção do vento. A opção mais fácil é a de um vento de um metro por segundo em uma direção de zero grau. A partir da nossa opção, a linha 5060 calcula a velocidade do vento na direção frontal do aparelho, e a linha 5070 calcula a componente lateral dessa velocidade. **SIN** e **COS** servem para calcular as componentes em cada direção. **WX** e **WY** são usadas para alterar a posição do aparelho — GX e GY na linha 5080.

A "alma" do programa é constituída pelas linhas 5500 a 5530, que chamam todas as grandes sub-rotinas em seqüência; desse modo, a posição é modificada continuamente.

A linha 5510 verifica, por intermédio do valor de **PZ**, se o avião aterrissou. O programa vai então para o segmento que verifica se a aterrissagem foi satisfatória ou desastrosa.

A linha 6000 verifica se as inclinações lateral e frontal do avião estão fora de limites aceitáveis, ou se a velocidade é maior que oitenta metros por segundo. Se houver uma dessas condições, ocorrerá um acidente. O motivo pode ser uma asa muito baixa, excesso de velocidade ou simplesmente uma tentativa de aterrissagem fora da pista. O programa prosseguirá na linha 6030, que cuida dos acidentes. Ela traça várias linhas irradiando do centro do pára-brisa. À medida que essas rachaduras são desenhadas, o computador emite um som. Após o acidente, a linha 6040 provoca uma pausa antes que o piloto seja informado das más notícias pela linha 6050, que fornece também os detalhes do acidente.

Se o avião pousar corretamente, a linha 6010 verificará a posição da aterrissagem. Se pousar fora da pista, a linha 6060 informará sobre o fato. Se a velocidade era inferior a cinqüenta metros por segundo, a linha 6070 dirá ao piloto que houve um pouso bem-sucedido. As linhas 6080 e 6090 informarão se o acidente foi pequeno, apenas com feridos leves, ou se foi um desastre mais grave; neste caso, o jeito é procurar a caixa-preta.



Apague a linha 5505. Note que algumas linhas deste programa são modificações de linhas já existentes.

```
2 DEF USR1=&H41:DEFUSR2=&H44
5 GOTO 5000
10 SCREEN 2,2:KEY OFF:COLOR 15,
4,2
15 A=USR1(0)
50 ON KEY GOSUB 3100,3200
60 KEY(1) STOP:KEY(2) STOP
110 RETURN
3000 A=STICK(0)
3010 KEY(1) ON:KEY(2) ON
3030 IF A=1 THEN PT=PT+3:GOTO 3
070
3040 IF A=5 THEN PT=PT-3:GOTO 3
070
3050 IF A=7 AND RL>-30 THEN RL=
RL-3:GOTO 3070
3060 IF A=3 AND RL<30 THEN RL=R
L+3
3070 KEY(1) STOP:KEY(2) STOP:RE
TURN
3100 T2=TC:IF TC>.2 THEN TC=TC-
.2:RETURN
3200 T2=TC:IF TC<8.8 THEN TC=TC
+.2:RETURN
3900 LINE(X,Y)-(X+50,Y+8),2,BF
5020 SCREEN 0:KEY OFF:INPUT"Vel
ocidade do vento (1-50 m/s) ",X
0
5025 IF X0>50 OR X0<0 THEN 5020
5030 INPUT"Direção do vento (0-
359 graus) ";X1
5035 IF X1>359 OR X1<0 THEN 503
0
5040 X0=X0/3
5050 PRINT:PRINT "Velocidade do
 vento: ";INT(3*X0);"m/s":PRINT
"Direção do vento :";X1;" graus
"
5060 WY=X0*COS(X1*C):WX=-X0*SIN
(X1*C)
5070 FOR I=1 TO 1000:NEXT:GOSUB
 10
5400 A=USR2(0)
5500 GOSUB 3000:GOSUB 1000
5510 IF PZ<=0 THEN 6000
5520 GOSUB 2000
5530 GOTO 5500
6000 IF ABS(RL)>RT OR PT>TP OR
PT<0 OR VV>80 THEN 6100
6010 IF ABS(PX)>WR OR ABS(PY)>1
000 THEN 6200
6020 COLOR 15,4,4:SCREEN 0
6030 PRINT "Parabéns !":PRINT "
Aterrissagem bem sucedida.":GOT
O 6500
6100 LINE (0,0)-(RND(1)*128+64,
RND(1)*96+49),15:LINE-(255,191)
,15:LINE (255,0)-(RND(1)*128+64
,RND(1)*96+49),15:LINE-(0,191),
15
6110 FOR K=1 TO 1000:NEXT
6120 COLOR 15,4,4:SCREEN 0:PRIN
T "Lamentamos informar, que o v
ôo PJ26"
6130 PRINT "sofreu um grave aci
dente."
```

```
6140 PRINT "Não há sobrevivente
s ":GOTO 6500
6200 COLOR 15,4,4:SCREEN 0
6210 PRINT "Você pousou fora da
pista."
6220 IF VV<40 THEN PRINT "Como
sua velocodade era baixa,":PRIN
T "quase nada aconteceu.":GOTO
6500
6230 IF VV<80 THEN PRINT "Entre
mortos e feridos,":PRINT "salv
aram-se todos.":GOTO 6500
6240 PRINT "Ninguém sobrevive a
um pouso":PRINT "nesta velocid
ade "
6500 LOCATE 5,10:PRINT "VALORES
RELATIVOS AO POUSO"
6510 PRINT:PRINT "Velocidade do
ar = ";INT(VV);"m/s":PRINT "Di
stância = ";INT(SQR(PY*PY+PX*PX
));" m":PRINT "Inclinação front
al = ",PT
6520 PRINT "Inclinação lateral
= ";RL:PRINT "Rotação do motor
= ",INT(TC*1000); "r.p.m "
6530 PRINT "Desvio do eixo da p
ista = ";INT(PX);"m":PRINT "Dir
eção = ",AD;"graus"
6540 PRINT:PRINT "Quer voar nov
amente (S/N) ?"
6550 A$=INKEY$:IF A$<>"S" AND A
$<>"N" THEN 6550
6560 IF A$="N" THEN CLS:END
6570 RUN
```

O laço principal do programa fica nas linhas 5500 a 5530. A primeira coisa que ele faz é chamar uma sub-rotina na linha 3000 que permitirá o controle do avião. Essa sub-rotina usa o comando **STICK**, por meio do qual podemos pilotar com o teclado — ou com o joystick, se usarmos **STICK (1)**.

As linhas 3030 e 3040 controlam os movimentos de subida e descida, enquanto as linhas 3050 e 3060 determinam os movimentos laterais.

É necessário ainda que existam formas de acelerar e diminuir a rotação do motor. Isso é feito por intermédio das teclas de função do MSX. Nessa rotina serão utilizadas as duas primeiras teclas de função. Para tanto, foi incluída a linha 50 no início do programa; ela informa ao computador o que fazer quando uma das teclas de função for pressionada. A função responsável por isto é **ON KEY GOSUB**, que difere da função **ON "variável" GOSUB** em diversos aspectos. O programa não será desviado ao encontrar a linha 50; ele apenas ficará sabendo para onde ir no momento oportuno, guardando os endereços na memória.

Só quando o programa chegar à linha 3010 é que as teclas de função serão ativadas pelo comando **KEY (N) ON**, onde N é o número da tecla. A partir daí, se pressionarmos a tecla de função 1, o programa executará a sub-rotina da linha 3100, que desacelera o motor. Se a tecla de função pressionada for a 2, o motor será acelerado pela sub-rotina da linha 3200.

Observe que o computador interromperá o que estiver fazendo para acelerar ou desacelerar o motor, voltando em seguida para a tarefa anterior. Contudo, se o programa estiver fazendo os cálculos no momento em que pressionarmos uma das teclas de função, coisas estranhas podem acontecer. É que a primeira parte dos cálculos terá sido feita com o valor antigo da rotação do motor, e a segunda, com o valor novo.

A solução não está em desligar as teclas de função durante a execução dos cálculos, pois, neste caso, o programa só aceitaria mudanças na rotação do motor em determinados momentos.

O problema, contudo, pode ser resolvido. Para isso, é necessário que se digite o comando **KEY (N) STOP** nas linhas 60 e 3070. Depois de um comando desse tipo, o computador será capaz de detectar a tecla de função correspondente, mas o desvio não ocorrerá até que um comando do tipo **KEY (N) ON** seja encontrado. Dessa maneira, o programa "sabe" que uma tecla de função foi pressionada, e se lembrará de tal fato no momento adequado.

Algumas linhas iniciais também foram modificadas. A linha 2 define o endereço de duas rotinas em linguagem de máquina, que ficam na ROM. A primeira "desliga" temporariamente a tela, quando chamada pela linha 15, impedindo que o jogador veja o desenho que está sendo feito na tela. A segunda "liga" a tela novamente, quando chamada pela linha 5400, de forma que a cabine apareça instantaneamente.

A linha 5 desvia o programa para que o jogador informe a velocidade e a direção do vento, nas linhas que vão de 5020 a 5070. Isso é feito antes que o desenho da cabine de comando seja feito pela sub-rotina 10.

Se a linha 5510 perceber que o avião atingiu o solo, o programa prosseguirá na linha 6000. Ali serão verificados os valores das inclinações frontal e lateral, assim como as da velocidade do vento. Se todos eles estiverem dentro dos limites aceitáveis, a linha 6020 informará ao piloto que houve uma aterrissagem bem-sucedida.

Um pouso fora da pista não significa necessariamente uma tragédia. Se a velocidade do avião for inferior a 40 m/s, a linha 6210 dirá que os danos foram pequenos. Se ela estiver entre 40 e 80 m/s, a linha 6220 dará conta de que as perdas materiais foram pequenas, mas que há alguns feridos. Se a velocidade for maior que 80 m/s só nos restará procurar a caixa-preta: o desastre terá sido total.

As linhas 6500 a 6530 mostram os valores de todas as variáveis no momento do pouso. Estudando bem esses números, você aprenderá a fazer aterrissagens cada vez melhores.

Finalmente, as linhas 6540 a 6570 nos dão a opção de jogar novamente.



Antes de mais nada, apague a linha 5505. O programa a seguir completa a simulação de vôo. Use as teclas Z, X, P e L para jogar.

```
3000 K =  PEEK ( - 16384): POKE
  - 16368,0
3010  IF K = 193 AND TC > .2 TH
EN TC = TC - .2: GOTO 3070
3020  IF K = 209 AND TC < 8.8 T
HEN TC = TC + .2: GOTO 3070
3030  IF K = 208 THEN PT = PT +
 3: GOTO 3070
3040  IF K = 204 THEN PT = PT -
 3: GOTO 3070
3050  IF K = 218 AND RL >  - .0
05 THEN RL = RL - .001: GOTO 30
70
3060  IF K = 216 AND RL < .005
THEN RL = RL + .001: GOTO 3070
3070  RETURN
5020  HOME : TEXT : INPUT "VELO
CIDADE DO VENTO (1-50) M/S ";X0
5025  IF X0 > 50 OR X0 < 1 THEN
 5020
5030  INPUT "DIRECAO DO VENTO
(0-359) GRAUS ";X1
5035  IF X1 > 359 OR X1 < 0 THE
N 5030
5040 X0 = X0 / 3
5060 WY = X0 *  COS (X1 * C)
5070 WX =  - X0 *  SIN (X1 * C)
5500  GOSUB 3000: GOSUB 1000
5510  IF PZ <  = 0 THEN 6000
5520  GOSUB 2000
5530  GOTO 5500
5570  GOTO 6550
6000  IF ( ABS (RL) > RT) OR PT
 > TP OR PT < 0 OR AS > 80 THEN
 6100
6010  IF  ABS (PX) > WR OR  ABS
 (PY) > 1000 THEN 6200
6020  TEXT : HOME : PRINT "PARA
BENS !": PRINT "ATERRISSAGEM BE
M SUCEDIDA": GOTO 6500
6100  HCOLOR= 3: HPLOT 0,0 TO
RND (1) * 140 + 71, RND (1) * 9
6 + 49 TO 279,191: HPLOT 279,0
TO  RND (1) * 140 + 71, RND (1)
 * 96 + 49 TO 0,191
6110  FOR K = 1 TO 2000: NEXT
6120  TEXT : HOME : PRINT "LAME
NTAMOS INFORMAR QUE O VOO PJ25"
: PRINT "SOFREU UM GRAVE ACIDEN
TE ": PRINT "NAO HA SOBREVIVENT
ES.": GOTO 6500
6200  TEXT : HOME : PRINT "VOCE
 POUSOU FORA DA PISTA."
```

```
6210  IF AS < 40 THEN  PRINT "C
OMO SUA VELOCIDADE ERA BAIXA, "
: PRINT "NADA ACONTECEU" GOTO 6
500
6220  IF AS < 80 THEN  PRINT "E
NTRE MORTOS E FERIDOS": PRINT "
SALVARAM-SE TODOS": GOTO 6500
6230  PRINT "NINGUEM SOBREVIVE
A UM POUSO FORCADO NESTA VELOCI
DADE"
6500  HTAB 5: VTAB 10: PRINT "V
ALORES RELATIVOS AO POUSO"
6510  PRINT : PRINT "VELOCIDADE
 DO AR "; INT (AS);" M/S": PRIN
T "DISTANCIA = "; INT ( SQR (PY
 * PY + PX * PX));" M": PRINT "
INCLINACAO FRONTAL = ";PT
6520  PRINT "INCLINACAO LATERAL
 = ";RL: PRINT "ROTACAO DO MOTO
R = "; INT (1000 * TC);" RPM"
6530  PRINT "DESVIO DO EIXO DA
PISTA = ";PX;" M": PRINT "DIREC
AO = ";AD;" GRAUS"
6540  PRINT : PRINT "QUER VOAR
NOVAMENTE (S/N)?"
6550  GET A$: IF A$ = "S" THEN
 RUN
6560  IF A$ = "N" THEN  HOME :
END
```

O laço principal do programa vai da linha 5500 à linha 5530. Inicialmente, ele chama a sub-rotina de controle do aparelho pelo teclado, que começa na linha 3000 e vai até a 3070. A linha 3000 verifica se alguma tecla foi apertada; as linhas 3010 e 3020 detectam as teclas Q e A, que diminuem e aceleram a rotação do motor; as linhas 3030 e 3040 descobrem as teclas P e L, que controlam a ascensão e a queda do aparelho; finalmente, as linhas 3050 e 3060 detectam as teclas Z e X, que viram o avião.

As linhas 5020 e 5070 permitem variar o nível de dificuldade da aterrissagem por intermédio da escolha da força e da inclinação do vento. A opção mais fácil é uma velocidade de 1 m/s numa direção que faz um ângulo de zero grau com o aparelho. A partir dos valores relacionados, a linha 5060 calcula o componente frontal da velocidade do vento, e a linha 5070, seu componente lateral. **WX** e **WY** são usados para alterar a posição do aparelho.

Se a linha 5510 perceber que o avião atingiu o solo, o programa prosseguirá na linha 6000. Ali serão verificados os valores das inclinações frontal e lateral, assim como os da velocidade do vento. Se todos eles estiverem dentro dos limites aceitáveis, a linha 6020 informará ao piloto que houve uma aterrissagem bem-sucedida.

Como no caso anterior, nem tudo estará perdido se aterrissarmos fora da pista. Caso a velocidade de pouso seja inferior a 40 m/s, a linha 6210 dirá que os danos foram pequenos. Se ela estiver entre 40 e 80 m/s, a linha 6220 dará conta de pequenos danos e alguns feridos. Se a velocidade for maior que 80 m/s, só nos restará procurar a caixa-preta entre os escombros: neste caso, o desastre terá sido total.

As linhas 6500 a 6530 mostram os valores de todas as variáveis no momento da aterrissagem. Estudando bem esses números, você aprenderá a pousar cada vez melhor.

As linhas 6540 a 6570, finalmente, nos dão a opção de jogar novamente.

Na última seção vimos que, se o jogo se estender por muito tempo, a tela gráfica pode ser corrompida por restos de cordões que se acumulam na memória. Se você tiver esse tipo de problema, pode corrigi-lo, modificando a linha 2000 para:

```
2000 X =  FRE (0): HCOLOR= 0
```

O comando **FRE(0)** limpa a área de cordões, evitando que ela invada a tela gráfica. No entanto, ele torna o programa mais lento.

A velocidade do programa é comprometida principalmente pela execução da sub-rotina que escreve na tela gráfica. Como os valores que devem ser constantemente atualizados ficam na porção inferior da tela, podemos acelerar consideravelmente o programa se liberarmos as quatro linhas inferiores para texto (veja o artigo *Os Comandos PEEK e POKE* na página 261) e usarmos **HTAB**, **VTAB** e **PRINT** para escrever ali esses valores.



Aqui estão as modificações para que o programa funcione no TK-2000.

```
10   FOR I = 0 TO 6
20   READ A: POKE 768 + I,A
30   NEXT
40   DATA  32,67,240,141,32,3,96
3000  POKE 800,0: CALL 768:K =
 PEEK (800)
```

As quatro primeiras linhas colocam uma pequena rotina em código de máquina na memória do TK-2000. Quando chamada pela linha 3000, essa rotina permite que se faça leitura do teclado, com auto-repetição.

A baixa velocidade de execução do programa se deve aqui às características específicas do TK-2000.



Antes de executar o programa, apague a linha 5505.

```
3000 IF PEEK(337)=255 THEN RETU
RN
3010 IF PEEK(341)=251 AND TC> 2
 THEN TC=TC- 2
3020 IF PEEK(344)=254 AND TC<8.
8 THEN TC=TC+ 2
3030 IF PEEK(341)=247 THEN PT=P
T+1
3040 IF PEEK(342)=2[illegible] THEN PT=P
T[illegible]
3050 IF PEEK(343)=22[illegible] AND RL> 3
0 THEN RL=RL-1
3060 IF PEEK(344)=247 AND RL<30
 THEN RL=RL+1
3070 RETURN
5020 CLS INPUT"VELOCIDADE DO VE
NTO (1-50) M/S";X0
5025 IF X0>50 OR X0 1 THEN 5C[illegible]
5030 INPUT"DIRECA[illegible] VENT[illegible] (
359) GRAUS" X1
5035 IF X1 359 OR X1[illegible] THEN 50
[illegible]
5040 X0=X0/3
5050 PRINT PRINT"VE[illegible]IADE [illegible]
VENTO= ";3*X0,"M/S".PRINT"DIREC
A[illegible]",X1,"GRAUS"
5060 WY=X[illegible]*[illegible]XI*[illegible]
5070 WX[illegible]X[illegible]N X[illegible]
[illegible]SUB [illegible]SUB [illegible]
55[illegible] IF P[illegible] THEN 600[illegible]
[illegible]
[illegible]
[illegible] IF ABS [illegible]PT [illegible] PT[illegible]
PT[illegible] OR V[illegible] THEN 6[illegible]0[illegible]
[illegible] IF ABS PX[illegible]WR[illegible] ABS PY[illegible]
```



```
0 0 THEN 6200
6020 CLS PRINT" PARABENS      AT
ERRISSAGEM BEM   SUCEDIDA " GOT
O 6500
6100 LINE(0,0)-(RND(128)+63,RND
 96)+48 ,PSET LINE-(255,191),PS
ET LINE(255,0 -(RND(128)+63,RND
(96 +48,,PSET LINE -(0,191),PSE
T
6110 FOR K 1 TO 2000 NEXT
6120 CLS:PRINT" LAMENTAMOS INFO
RMAR QUE O JOJ   PJ26 SOFREU UM
 GRAVE ACIDENTE "
6140 PRINT" NAO HA SOBREVIVENTE
S" GOTO 6500
6200 CLS:PRINT" VOCE POUSOU FOR
A DA PISTA"
6210 IF VV<40 THEN PRINT" COMO
SUA VELOCIDADE ERA BAIXA,  QUAS
E NADA ACONTECEU" GOTO 6500
6220 IF VV<80 THEN PRINT" ENTRE
 MORTOS E FERIDOS,         SALV
ARAM-SE TODOS":GOTO 6500
6230 PRINT" NINGUEM SOBREVIVE A
 UM POUSO    NESTA VELOCIDADE!"
6500 PRINT:PRINT " VALORES RELA
TIVOS AO POUSO"
6510 PRINT:PRINT" VELOCIDADE DO
 AR =",INT(VV):"M/S":PRINT" DIS
TANCIA =",INT(SQR(PY*PY+PX*PX))
 PRINT" INCLINACAO FRONTAL =";P
T
6520 PRINT" INCLINACAO LATERAL
=":RL PRINT" ROTACAO DO MOTOR =
",INT(10*TC)/10," X 1000"
6530 PRINT" DESVIO DO EIXO DA P
ISTA =",INT(ABS(PX)):"M"·PRINT"
 DIRECAO =",AD,"GRAUS"
6540 PRINT PRINT" QUER VOAR NOV
AMENTE (S/N) ?"
6550 A$=INKEY$:IF A$<>"S" AND A
$<>"N" THEN 6550
6560 IF A$="N" THEN CLS·END
6570 RUN
```

O laço principal vai de 5500 a 5530. Ele chama a sub-rotina de controle do aparelho pelo teclado, que vai de 3000 a 3070. A linha 3000 verifica se alguma tecla foi apertada; 3010 e 3020 detectam as teclas F e S, que diminuem e aceleram a rotação do motor; 3030 e 3040 descobrem as setas "para cima" e "para baixo", que controlam a ascensão e a queda do aparelho; 3050 e 3060 detectam as setas "direita" e "esquerda", que viram o avião.

As linhas 5020 e 5070 permitem variar o nível de dificuldade do pouso, escolhendo a força e a inclinação do vento. A melhor opção é uma velocidade de 1 m/s numa direção que faz um ângulo de zero grau com o aparelho. A linha 5060 calcula o componente frontal da velocidade do vento, e a 5070, seu componente lateral. **WX** e **WY** servem para alterar a posição do aparelho.

Se a linha 5510 perceber que o avião atingiu o solo, o programa prosseguirá na linha 6000. Ali serão verificados os valores das inclinações frontal e lateral, bem como da velocidade do vento. Se todos eles estiverem dentro dos limites aceitáveis, a linha 6020 informará ao piloto que houve uma aterrissagem bem-sucedida.

Uma aterrissagem fora da pista não implica, necessariamente, a destruição do aparelho. Assim, se a velocidade de pouso for inferior a 40 m/s, a linha 6210 dirá que os danos foram pequenos. Se ela estiver entre 40 e 80 m/s, a linha 6220 informará que alguma coisa foi destruída e que há alguns feridos. Se a velocidade for maior que 80 m/s, só nos restará procurar a caixa-preta entre os destroços do avião: o desastre terá sido total.

As linhas 6500 a 6530 mostram os valores das variáveis no momento do pouso. Estudando bem esses números, você aprenderá a pousar cada vez melhor.

Finalmente, as linhas 6550 a 6570 oferecem a opção de jogar de novo.



UM RELÓGIO NA TELA

Se você não tiver nada para fazer numa tarde de chuva, rompa o tédio executando a rotina apresentada neste artigo e veja surgir um relógio digital na tela do computador.

Usado para regular as atividades do micro, o relógio interno funciona a uma velocidade constante. Em alguns casos, seu conteúdo só pode ser lido e manipulado por instruções em código de máquina. Em outros, podemos cronometrar eventos através de instruções do BASIC, que variam segundo o modelo da máquina: **PAUSE** no Spectrum, **TIME** no MSX, **TIMER** no TRS-Color etc.

Tais instruções fazem com que o micro conte em unidades entre cem e cinqüenta avos de segundo, conforme a velocidade do relógio. Muitas operações usam o relógio de maneira similar: por exemplo, se o micro for programado para tocar música, temos que especificar a duração de cada nota.

CONTANDO O TEMPO

De fato, ainda que certas funções não tenham sua duração definida dessa maneira, o computador é um contador de tempo por excelência, executando os programas com o olho no relógio. Podemos, assim, fazer com que o computador conte o tempo para nós: basta escrever um programa em BASIC contendo um laço que imprima a hora, provoque uma pausa, imprima a nova hora, e assim sucessivamente. Se fizermos isso, logo veremos que a pausa deve ser um pouco menor que um segundo, pois o micro leva algum tempo para fazer as adições e impressões necessárias.

Um relógio desse tipo tem duas grandes desvantagens. A primeira é que ele só conta o tempo enquanto o computador estiver ligado. Mais grave do que esta, a segunda limitação consiste no fato de que, sempre que quisermos usar o micro para alguma outra coisa, teremos que desligar o relógio, pois não é possível executar dois programas BASIC ao mesmo tempo. O "desligamento" do relógio caracteriza, portanto, uma interrupção da UCP. Esse tipo de manipulação em geral exige um conhecimento maior da arquitetura do microcomputador, bem como o uso de instruções de acesso a recursos que só existem em linguagem de máquina.

UM RELÓGIO INTERMITENTE

Para que um programa funcione simultaneamente com a execução de um BASIC, é preciso que ele contenha rotinas controladas por interrupção.

Os usuários do Spectrum já tiveram um exemplo desse tipo de rotina. De fato, o método utilizado é semelhante em todos os micros.

Quando está ligado, o micro é constantemente interrompido por uma fração de segundo, a intervalos regulares. Isso acontece mesmo que um programa em BASIC esteja sendo executado, pois o computador precisa "saber" se alguma tecla foi pressionada. Assim, o programa BASIC é interrompido enquanto o teclado é verificado, recomeçando novamente até a interrupção seguinte.

Durante a verificação do teclado, é possível introduzir uma rotina em código, de modo a executá-la nas "brechas" abertas pela interrupção do BASIC. O resultado é que os programas parecerão funcionar simultaneamente.



- O COMPUTADOR COMO RELÓGIO
- CONTANDO O TEMPO
- COMO FAZER INTERRUPÇÕES
- O MOSTRADOR DO RELÓGIO
- UM RELÓGIO INTERMITENTE
- UMA ROTINA EM CÓDIGO
- LIGUE O RELÓGIO
- ACERTE OS PONTEIROS
- SAVE, LOAD, SOUND, PLAY

Já que o processo de interrupção é controlado pelo relógio do micro, convém acertar o passo do relógio simplesmente contando o número de interrupções. A frequência de interrupções varia de micro para micro, mas o princípio é o mesmo.

O programa que se segue simula um relógio no Spectrum e no TRS-Color, com horas, minutos e segundos contados a partir do momento em que começa a ser executado. Podemos acertá-lo de modo a fazê-lo funcionar, seja como relógio, seja como cronômetro. Esses valores são mostrados continuamente no canto superior direito da tela. O relógio permanecerá funcionando na tela, mesmo que outro programa esteja sendo executado ou digitado.

Como você não tardará a perceber, ele não prima muito pela precisão. Essa falha, porém, não se deve à rotina que acerta a hora, pois os atrasos provocados por ela são insignificantes (da ordem de milionésimos de segundo). Assim, o erro vem mais do relógio interno do que de qualquer outra fonte. Por outro lado, funções como **SAVE, LOAD, SOUND** e **PLAY** param o relógio enquanto estiverem sendo executadas. A hora é exibida, num mostrador digital, no canto superior direito do vídeo, apagando qualquer coisa impressa ali anteriormente. Se isso causar problema, reorganize sua tela a fim de evitar a primeira linha. Por exemplo, você poderia colocar a hora digital no canto superior (esquerdo ou direito), desde que o modelo do seu micro o permitisse. Neste caso, o programa em linguagem de máquina teria que ser alterado.



O programa a seguir funciona tanto na versão de 16K, como na de 48K do Spectrum. Antes de começar a rodá-lo, tome alguns cuidados especiais, pois cartuchos e expansões conectados ao micro podem dificultar sua execução. É o caso, por exemplo, de interfaces para impressora, joysticks, etc.

```
10 CLEAR 32319· LET total=0
20 FOR n=32320 TO 32554: READ
a  POKE n,a: LET total=total+
a  NEXT n
30 IF total<>24216 THEN
PRINT "Erro nos dados"  STOP
40 RAND USR 32320
50 DATA 33,0,0,34,120,92.34.
121,92,62,40,237,71.237.94.
201,0.64.0.0
60 DATA 62,62,237.71.237.86,
201,0,229,213.197.245,58,91,
126,60,50,91.126.254
70 DATA 50.32.50.175.50.91.
126,58.120,92,60.50.120,92,
254,60,32,35.175,50
80 DATA 120,92.58.121.92,60.
50.121,92,254.60,32,20,175.50
,121,92,58,122,92
90 DATA 60,50.122.92,254,13,
32.5.62, ,50,122,92.58.122,92
.38,0,111,17
100 DATA 23,64,205,234,126,58,
121.92.38,0,111,17,76,64,205,
234.126,58,120,92
110 DATA 38.0,111,17.29.64.205
,234,126,17,208.61.33.29,64,
205,34,127,17,208
120 DATA 61,33,26,64,205,34,
127.62.120,33,24,88.119.17.25,
88,1,7,0,237
130 DATA 176,205,191.2.241.193
.209,225.251.201,237,83,80.126
.1,246,255.205,251.126
140 DATA 1,255,255,205,251,126
.201,175,9,60.56,252.237,66,61
,198,48,229,205,21
150 DATA 127,33,80,126,52,42.
80,126,205,34,127,225,201,237,
75,54.92,38,0,111
160 DATA 41,41,41,9,235,201,6,
8,26,119,36,19,16,250,201
```

Essa rotina é formada por números distribuídos em linhas **DATA**; eles são colocados na memória pelo BASIC, usando **POKE**. Tal quantidade de números propicia erros de digitação; assim, a linha 20 verifica a soma dos números: se o resultado não corresponder ao esperado, a linha 30 interromperá a execução do programa com uma mensagem de erro, para que você verifique as linhas **DATA**. Esse método é chamado de "checagem por soma" e é recomendável quando há muitos números.

A linha 10 protege o topo da memória para colocar ali a rotina em código, que é executada pela linha 40 . O relógio começa com 00:00:00, mas podemos acertá-lo com:

```
POKE 23672, (segundos)
POKE 23673, (minutos)
POKE 23674, (horas)
```

Os números nos comandos **POKE** devem estar no intervalo permitido — de 0 a 60 para minutos e segundos, e de 1 a 12 para horas. Se quisermos "zerar" o relógio, é mais fácil usar:

```
RAND USR 32320
```

que executa a rotina novamente. Você vai precisar desse comando, se o programa em BASIC for apagado por **NEW**.



Essa rotina não funciona com drives de disquete ligados ao micro.

```
10 CLEAR 200,32599
20 FOR J=32600 TO 32679
30 READ N
40 POKE J,N
50 NEXT
100 DATA 204,0,0,253,127,252,25
3,127,254,48,140,4,191,1,13,57
110 DATA 206,127,164,142,128,0,
166,130,76,161,192,38,9,111,132
120 DATA 140,127,252,38,242,134
,1,167,132,206,4,32,142,127,255
130 DATA 79,230,130,192,10,45,3
,76,32,249,195,47,58,237,195,17
140 DATA 131,4,25,47,6,134,58,1
67,194,32,229,126,137,76,50,60,
60,13
```

A rotina em código está nas linhas **DATA**, sendo colocada na memória pelo **POKE** da linha 40. Qualquer erro nas linhas **DATA** será fatal; portanto, grave o programa antes de executá-lo. Verifique os números cuidadosamente. Um erro muito comum é esquecer alguma vírgula, substituir uma vírgula por um ponto ou vice-versa. Tudo isto prejudica a sequência de leitura do comando **READ**. Execute o programa para carregar a rotina em código. Para ligar o relógio, use:

```
EXEC 32600
```

Isso inicia a atividade do relógio com 00:00:00. Para acertá-lo, digite em modo direto, pelo teclado:

```
POKE 32764, (horas)
POKE 32765, (minutos)
POKE 32766, (segundos)
```

Não empregue valores impróprios para horas, minutos e segundos: o computador não tem meios de checá-los.

SPRITES PARA O TRS-80 (2)

- CORRESPONDÊNCIA ENTRE CÓDIGO E CARACTERE GRÁFICO
- COMO CALCULAR O CÓDIGO
- BITS QUE SE ACENDEM
- APLICAÇÕES

Aprenda a empregar os caracteres gráficos pré-programados do TRS-80 em desenhos mais complexos e capacite-se a trabalhar com os úteis "sprites programáveis".

Se examinarmos com cuidado os caracteres gráficos do TRS-80, que ocupam a faixa de codigos que vai de 129 a 193 (veja tabela no capítulo anterior desta lição), podemos notar facilmente que eles são formados por todas as combinações possíveis de seis *pixels* retangulares, acesos ou apagados. Portanto, a matriz de um caractere tem esta forma:



Você verá agora que não é necessário memorizar a tabela de caracteres gráficos do TRS-80, nem consultá-la a todo momento ao se projetar um desenho mais complexo, formado por vários desses caracteres. A razão é que existe uma relação matemática bem definida entre a forma do gráfico e o seu código numérico. Ou seja, os códigos não foram atribuídos ao acaso, como para computadores de outras linhas.

BITS QUE SE ACENDEM

Todo caractere gráfico é formado pela combinação de seis pixels — dois na horizontal e três na vertical —, que podem estar individualmente acesos ou apagados. Cada um deles corresponde exatamente ao pontinho de luz que pode ser ligado ou desligado com os comandos gráficos **SET** e **RESET**.

Por quê? Sabemos que a tela de baixa resolução do TRS-80 tem 128 pontos na horizontal (ou seja, 64 caracteres vezes 2 pixels), e 48 na vertical (isto é, 16 linhas vezes 3 pixels).

Cada pixel corresponde a um bit na memória de vídeo do micro. Como cada byte da memória tem oito bits, e cada caractere seis bits, sobram dois bits não usados para representar pixels. Um byte da memória de vídeo no TRS-80 poderia ser representado assim:

0	1
2	3
4	5

7	6	5	4	3	2	1	0

Os bits numerados de 0 a 5 correspondem aos seis pixels habituais do caractere gráfico.

O bit 6 não é usado para representação de caracteres gráficos, e o bit 7 é sempre igualado a 1, para todos os caracteres gráficos.

Por exemplo, se quisermos obter o código gráfico 134, precisaremos acender sucessivamente 1 e 2:

7	6	5	4	3	2	1	0
1		0	0	0	1	1	0

O código 134 é obtido fazendo-se a conversão de binário para decimal:

bit	valor		multiplicador		resultado
0	0	x	1	=	0
1	1	x	2	=	2
2	1	x	4	=	4
3	0	x	8	=	0
4	0	x	16	=	0
5	0	x	32	=	0
6	0	x	64	=	0
7	1	x	128	=	128
Soma					134

Dessa maneira, é relativamente fácil calcular o valor do código de qualquer caractere. Para isso, basta verificar as posições ocupadas pelos diversos pixels "acesos" na matriz do caractere, adicionar em seguida os valores inscritos na posição, e somar 128 ao resultado (daí a razão de o número 128 ser considerado também como um código gráfico):

1	2
4	8
16	32

APLICAÇÕES

Este fato tem diversas implicações: é muito fácil, por exemplo, inverter-se os gráficos escritos na tela (para isso, basta transformar os bits da memória gráfica, de modo que os que são 1 passem para 0, e vice-versa). É possível obter outras transformações matemáticas com os códigos gráficos, com resultados curiosos.

Mas a aplicação que mais nos interessa no momento relaciona-se com a definição de figuras complexas. Neste caso, o primeiro passo consiste em armazenar os códigos numéricos dos caracteres gráficos que compõem tais figuras, em uma linha **DATA**, que é lida e armazenada (concatenada) em uma variável alfanumérica. Para economizar espaço, somente o valor diminuído de 128 é armazenado.

Tomemos como exemplo o programa a seguir; ele define uma variável **A$** que contém a imagem de um aeroplano simples, formado pelos caracteres 141, 170, 174, 170 e 140.

```
10 CLS:A$=""
20 FOR I=1 TO 5:READ N
30 A$=A$+CHR$(128+N)
40 NEXT I
50 DATA 13,42,46,42,12
```

O aviãozinho "voa" do seguinte modo: impresso inicialmente em uma posição da tela (com **PRINT @**), ele é sucessivamente apagado com cinco caracteres em branco e exibido na posição seguinte, e assim por diante. Acrescente estas linhas e rode o programa:

```
60 FOR I=256 TO 313
70 PRINT @ I,A$;
80 FOR J=1 TO 40:NEXT
90 PRINT @ I,"     ";
100 NEXT I
```

Portanto, **A$** é, de certa forma, uma espécie de sprite.

GRÁFICOS DA ROM NO SPECTRUM (2)

- CARACTERES GRÁFICOS DENTRO DO PROGRAMA
- A FUNÇÃO CHR$
- LINHAS LONGAS
- A FUNÇÃO TO

Em artigo anterior, explicamos o que são os caracteres gráficos e como entrá-los diretamente pelo teclado. Trataremos aqui de sua utilização dentro de programas.

Para especificar caracteres gráficos dentro de um programa, sem precisar digitá-los diretamente (veja artigo publicado à página 640), emprega-se a utilíssima função **CHR$**. Para imprimir na tela um quadrado vermelho, podemos usar, por exemplo:

```
PRINT INK 2,CHR$(143)
```

O programa seguinte mostra a tabela de correspondência entre códigos numéricos e gráficos na tela dos micros da linha Spectrum:

```
10 CLS
20 FOR j=128 TO 143 STEP 5
30 FOR i=j TO j+4
40 PRINT i;" ";CHR$(i);" ";
50 NEXT i
60 PRINT
70 NEXT j
```

Os caracteres são impressos ordenadamente em fileiras, na tela, por meio dos dois laços que começam nas linhas 20 e 30 do programa. O primeiro laço varia J de 193 a 242 (a faixa de códigos correspondente aos caracteres gráficos), de 6 em 6. O laço seguinte percorre todos os valores numéricos existentes entre **J** e **J+5**.

O **PRINT** da linha 60 serve para encerrar uma fileira de seis códigos e suas representações gráficas. Estas são exibidas pela linha 40.

LINHAS LONGAS

Se você pretende empregar caracteres gráficos com certa frequência em um programa, convém armazenar seus códigos em uma variável alfanumérica. Assim, ficará bem mais fácil utilizá-los, e você não terá necessidade de consultar a todo momento a tabela de códigos gráficos, quando estiver desenvolvendo o programa

Da mesma maneira, quando se pretende utilizar uma cadeia de caracteres gráficos em vários pontos de um programa, é interessante armazená-los em uma variável alfanumérica. Como o Spectrum não tem a função **STRING$**, que permite fazer isto com um único comando, será preciso escrever um pequeno laço de acumulação:

```
10 CLS
20 PRINT AT 19,0;"Codigo
Grafico (128-143) : "
25 INPUT c
30 PRINT AT 19,0;"Comprimento
(1-29) : "
40 INPUT n
50 LET s$=""
60 FOR i=1 TO n
70 LET s$=s$+CHR$(c)
80 NEXT i
90 PRINT AT 10,1;s$
95 PAUSE 100
100 GOTO 10
```

A cadeia alfanumerica **S$** tem **L** códigos, que são exibidos na tela ao mesmo tempo, quando se emprega o comando **PRINT S$**. Este truque será muito útil quando você precisar usar com freqüência, no programa, linhas de separação, molduras etc.

A FUNÇÃO TO

A função **TO** tem grande utilidade na construção de gráficos com símbolos de teclado, pois ela permite que se extraiam segmentos curtos de uma cadeia gráfica maior.

Suponhamos que você precise usar retas horizontais de diferentes tamanhos, na composição de uma tela gráfica. Em vez de gerar várias cadeias gráficas, usando a técnica do laço **FOR...NEXT** explicada anteriormente, você poderá gerar uma única cadeia, com comprimento máximo (uma cadeia **C$**, com 32 caracteres, por exemplo).

Depois, para colocar em um ponto qualquer da tela do micro uma cadeia gráfica de doze caracteres, bastará utilizar o **PRINT AT** combinado com **C$(TO 12)**. A função **TO**, no caso, serve para extrair os primeiros doze caracteres da cadeia **C$**, simplificando muito o processo.

MICRO DICAS

CARACTERES INVERTIDOS

Em alguns tipos de computador, o conjunto de caracteres gráficos — isto é, aqueles caracteres que podem ser obtidos pressionando-se **<SHIFT><9>** — abrange também os chamados caracteres invertidos

Estes nada mais são do que uma representação "em negativo" da forma usual de exibição na tela. As duas formas — "normal" e "invertida" — aparecerão diferentemente conforme a marca do computador.

Nos micros originais da Sinclair (o ZX 81 e seu equivalente norte-americano, o Timex), por exemplo, a representação normal de vídeo compõe-se de caracteres pretos sobre fundo claro O caractere invertido será, portanto, branco sobre fundo preto. Como o fundo normal de tela é branco, os caracteres invertidos aparecem como um retângulo preto, com o caractere impresso em branco. Já a cor clara do fundo dos caracteres em vídeo normal fica igual ao fundo da tela. Esse tipo de representação de vídeo é característico dos modelos nacionais da Microdigital (TK-82C, TK 83 e TK-85)

Alguns outros computadores nacionais, entre eles o CP-200, da Prológica, têm tela com fundo preto, normalmente, e exibem caracteres claros Aqui, o caractere invertido, obtido com **<SHIFT><9>**, é escuro sobre fundo claro.

Existem, ainda, alguns modelos que permitem ao usuário mudar de vídeo preto para branco, conforme sua conveniência, por meio de um interruptor colocado no console.

Os caracteres invertidos têm diversas aplicações. São utilizados, particularmente, para realçar determinados títulos, rótulos ou mensagens numa tela. Alguns deles — como ponto, dois pontos, asterisco, barra e a letra O — podem ser incorporados como elementos gráficos na composição de um desenho.

UM JOGO DE ESTRATÉGIA

Entre no mundo dos grandes riscos — e grandes lucros. Você é dono de uma empresa de mineração. Terá astúcia e talento para tomar decisões e coragem para levá-las a cabo?



Neste jogo de estratégia empresarial, o jogador desempenha o papel de dono de uma companhia de mineração. Seu dever é cuidar para que o empreendimento prospere ao máximo. Como durante o jogo muitas opções lhe são oferecidas, a sorte da empresa dependerá de sua capacidade de tomar decisões.

Jogos de estratégia, assim como de aventuras, são geralmente escritos em BASIC, não havendo necessidade de se recorrer a rotinas em código. Como eles também não incluem longos textos e comentários, são compatíveis com micros de pouca quantidade de memória. Embora nosso jogo tenha sido enriquecido com rotinas gráficas que ilustram o processo de mineração — o que aumentou o espaço de memória utilizado —, o programa cabe nos micros de 16K.

Apresentaremos o programa em duas partes, pois ele é bastante longo. Trataremos aqui de sua parte central, mas algumas das rotinas essenciais para fazê-lo funcionar só serão abordadas no próximo artigo.

Depois de digitar esta primeira seção, grave o programa e aguarde a publicação da parte que o completa. Em alguns computadores, se o programa for executado neste estágio, a tela será preenchida com diversas informações sobre o *status* do jogo, além de uma série de opções. Contudo, surgirá também uma mensagem de erro, uma vez que o programa está incompleto.

OS OBJETIVOS DO JOGO

Quando o jogo começa, o jogador tem um saldo a seu favor — a companhia de mineração e 2 milhões em dinheiro. Sua tarefa é investir estes recursos inteligentemente, na busca do precioso metal. O objetivo do jogo é reunir a maior quantia de dinheiro possível em trinta rodadas. O jogo permite que uma pessoa jogue sozinha ou que duas joguem ao mesmo tempo.

A cada lance, o jogador se vê diante de várias opções. Antes de começar as escavações, precisa encontrar um local promissor; para isso, deverá destinar certa quantia à contratação de um geólogo que lhe forneça uma avaliação profissional. O jogador será, então, informado sobre suas chances de encontrar ouro e da provável profundidade e dimensões do veio. A responsabilidade de decidir se vale a pena explorar determinada mina é sua.

Como o trabalho de escavação e garimpagem envolve altas somas, poderá ser conveniente investir na pesquisa e desenvolvimento de equipamentos que reduzam os custos da operação. Ou talvez seja melhor começar logo as escavações — só o jogador pode decidir. Se ele iniciá-las, gráficos coloridos ilustrarão o progresso do trabalho. Caso não encontre ouro, precisará resolver se vale a pena continuar a prospecção ou se é melhor abandonar a mina e partir para outra.

No decorrer do jogo, dois outros fatores influem nos resultados. Uma vez encontrado o ouro, pode-se guardá-lo em cofre-forte ou vendê-lo no mercado. É razoável guardar o metal, se não houver necessidade imediata de dinheiro; pode ser vantajoso esperar por uma boa cotação para vendê-lo — a cotação flutua durante o jogo. Contudo, armazenar ouro é arriscado, pois há ladrões por toda parte e, quanto maior a quantidade, maior a tentação.

A segunda parte do programa cuidará dos detalhes do funcionamento do jogo. Agora, digite a primeira parte.

```
 5 BORDER 6: PAPER 6: INK 0:
 CLS
10 PRINT AT 9,2;"Quantos joga
dores? (1 ou 2)": LET a$=
INKEY$: IF a$="" THEN  GOTO
10
20 IF a$<"1" OR a$>"2" THEN
GOTO 10
30 LET p=VAL a$: LET nop=p
40 DIM a(2,6): DIM c(2,5):
DIM a$(p,8): DIM r(2): LET er
=10000
50 LET r(1)=0: LET r(2)=0:
LET a(1,1)=2000000: LET a(1,2
)=2000000: LET a(2,1)=2000000
: LET a(2,2)=2000000: LET a(1
,3)=0: LET a(2,3)=0: LET a(1,
4)=100000: LET a(2,4)=100000:
 LET a(1,5)=0: LET a(2,5)=0:
LET a(1,6)=0: LET a(2,6)=0:
```

- OBJETIVOS DO JOGO
- UM OU DOIS JOGADORES
- A PRIMEIRA TELA
- SALDOS E CUSTOS
- COMO DAR INFORMAÇÕES AO PROGRAMA
- OFERTA DE OPÇÕES AO JOGADOR
- ROTINA DOS LADRÕES DE OURO
- EFEITOS SONOROS





```
PRINT
70 FOR n=1 TO p: INPUT "Nome
do jogador ";(n);"?", LINE a$
(n): NEXT n
200 FOR n=1 TO 30: FOR m=1 TO
nop
202 BORDER 7: PAPER 7: INK 0:
CLS
210 PRINT  PAPER 6;n: PRINT
PAPER 1; INK 6;AT 0,4;" M I N
A   D E   O U R O "
220 PRINT 'TAB 16;a$(1);: IF n
op=2 THEN  PRINT TAB 24;a$(2);
230 PRINT '"SALDO TOTAL  $";
TAB 15;a(1,1);: IF nop=2 THEN
PRINT TAB 24;a(2,1);
240 PRINT '"DINHEIRO     $";
TAB 15;a(1,2);: IF nop=2 THEN
PRINT TAB 24;a(2,2);
250 PRINT '"OURO kg";TAB 15;a(
1,3);: IF nop=2 THEN  PRINT
TAB 24;a(2,3);
260 PRINT '"CUSTO P/CAVAR$";
TAB 15;a(1,4);: IF nop=2 THEN
PRINT TAB 24;a(2,4);
270 PRINT '"No. DE MINAS";TAB
15;a(1,5);: IF nop=2 THEN
PRINT TAB 24;a(2,5);
280 PRINT '"PROFUNDID. m";TAB
15;a(1,6);: IF nop=2 THEN
PRINT TAB 24;a(2,6),
300 PRINT '' PAPER 4; INK 0;"C
otacao do ouro no mercado:":
PRINT "$";er;" por kg de ouro"
400 PRINT ' PAPER 5;">-";a$(m)
500 PRINT  PAPER 2; INK 7;"1";
: PRINT "-Pesquisa e desenvolv
imento"
510 PRINT  PAPER 2; INK 7;"2";
: PRINT "-Levantamento geologi
co"
520 PRINT  PAPER 2; INK 7;"3";
: PRINT "-Cavar mais 200 m"
530 PRINT  PAPER 2; INK 7;"4";
: PRINT "-Vender ouro no merca
do"
540 PRINT  PAPER 2; INK 7;"5";
: PRINT "-Passa a vez"
550 PRINT : PRINT  FLASH 1;
PAPER 1; INK 6;"Faca sua opcao
"
600 LET i$=INKEY$: IF i$=""
THEN  GOTO 600
610 IF i$<"1" OR i$>"5" THEN
GOTO 600
620 GOSUB VAL i$*1000
700 IF a(m,2)<0 THEN  GOTO
 7000
710 LET er=er+INT (RND*1000)-
 200
720 IF INT (RND*1600)-a(m,3)<0
THEN  GOSUB 900
740 LET a(m,1)=a(m,2)+a(m,3)*
er
750 PAPER 7: INK 0: BORDER 7:
CLS
790 NEXT m
800 NEXT n
810 PAPER 5: BORDER 5: INK 0:
CLS
820 PRINT  FLASH 1; INK 7;
PAPER 2;AT 6,10;" FIM DO JOGO"
```

```
830 PRINT 'p5;"Saldo total de
";a$(1): PRINT TAB 11;"$";a(1,
1)
840 IF nop=2 THEN  PRINT 'TAB
5;"Saldo total de ";a$(2):
PRINT TAB 11;"$";a(2,1)
850 PRINT '' PAPER 2; INK 6;
FLASH 1,TAB 1;"Qualquer tecla
para recomecar"
860 IF INKEY$<>"" THEN  GOTO
 860
870 IF INKEY$="" THEN  GOTO
 870
880 RUN
900 PAPER 2: INK 6: BORDER 2:
 CLS
905 LET jk=INT (RND*100)+50:
IF jk>a(m,3) THEN  LET jk=a(m,
3)
910 PRINT  PAPER 6; INK 1;
FLASH 1;AT 9,10;" R O U B O "
920 PRINT : PRINT  INK 7;"
Foram roubados ";jk;"kg de":
PRINT "     seu ouro": LET a(m,
3)=a(m,3)-jk: LET a(m,1)=a(m,1
)-(jk*er)
930 FOR x=1 TO 35: SOUND  .05,
40: SOUND .05,20: NEXT x
940 BORDER 7: PAPER 7: INK 0:
CLS : RETURN
```

```
20 DIM A(1,5),C(1,4)
90 SCREEN 1:COLOR 1,10,4:KEY OF
F
100 VPOKE BASE(6),4*16+4:VPOKE
BASE(6)+1,6*16+6
110 FOR I=6 TO 7:VPOKE BASE(6)+
I,15*16+6:NEXT
120 LOCATE 0,5:PRINT "Quantos j
ogadores (1 ou 2) ?;
130 A$=INKEY$:IF A$<"1" OR A$>"
2" THEN 130
140 P=VAL(A$):NO=P:ER=10000:A(0
,0)=2000000#:A(0,1)=A(0,0):A(1,
0)=A(0,0):A(1,1)=A(0,0):A(0,3)=
100000!:A(1,3)=A(0,3)
150 FOR N=1 TO P:PRINT:PRINT:PR
INT"Nome do jogador";N;:LINE IN
PUT A$(N-1):IF LEN(A$(N-1))>8 T
HEN A$(N-1)=LEFT$(A$(N-1),8)
160 NEXT
200 FOR N=0 TO 29:FOR M=0 TO NO
-1
202 FOR I=0 TO 767:VPOKE BASE(5
)+I,0:NEXT:LOCATE 0,0
204 FOR I=0 TO 31:VPOKE BASE(5)
+I,8:VPOKE BASE(5)+736+I,8:NEXT
206 FOR I=0 TO 24:VPOKE BASE(5)
+I*32,8:VPOKE BASE(5)+31+I*32,8
:NEXT
210 LOCATE 9,0:PRINT "MINA DE O
URO"
220 LOCATE 12,2:PRINT A$(0);:IF
 NO=2 THEN LOCATE 21,2:PRINT A$
(1)
230 LOCATE 0,3:PRINT "SALDO TOT
AL";TAB(11);A(0,0);:IF NO=2 THE
N LOCATE 20,3:PRINT A(1,0)
240 LOCATE 0,4:PRINT "DINHEIRO"
;TAB(11);A(0,1);:IF NO=2 THEN L
OCATE 20,4:PRINT A(1,1)
250 LOCATE 0,5:PRINT "OURO";TAB
(11);A(0,2);:IF NO=2 THEN LOCAT
E 20,5:PRINT A(1,2)
260 LOCATE 0,6:PRINT "$ P/ CAVA
R";TAB(11);A(0,3);:IF NO=2 THEN
 LOCATE 20,6:PRINT A(1,3)
270 LOCATE 0,7:PRINT "Nº DE MIN
AS";TAB(11);A(0,4);:IF NO=2 THE
N LOCATE 20,7:PRINT A(1,4)
280 LOCATE 0,8:PRINT "PROFUNDI/
/";TAB(11);A(0,5);:IF NO=2 THEN
 LOCATE 20,8:PRINT A(1,5)
300 LOCATE 1,10:PRINT "COTAÇÃO
DO OURO NO MERCADO":LOCATE 3,11
:PRINT ER;"POR KG DE OURO"
400 LOCATE 12,13:PRINT A$(M)
500 PRINT:PRINT "1-Pesquisa e d
esenvolvimento"
510 PRINT "2-Levantamento geoló
gico      "
520 PRINT "3-Cavar mais 200 m
          "
530 PRINT "4-Vender ouro no mer
cado      "
540 PRINT "5-Passa a vez
          "
600 A$=INKEY$:IF A$<"1" OR A$>"
5" THEN 600
620 ON VAL(A$) GOSUB 1000,2000,
3000,4000,5000
700 IF A(M,1)<0 THEN 7000
710 ER=ER+1000*INT(RND(1))-200
720 IF RND(1600)-A(M,2)<0 GOSUB
 900
740 A(M,0)=A(M,1)+A(M,2)*ER
750 CLS
790 NEXT M,N
810 CLS
820 LOCATE (5,5):PRINT "FIM DE
JOGO"
830 LOCATE (0,7):PRINT "SALDO T
OTAL DE";A$(0);TAB(23);A(0,0)
840 IF NO=2 THEN PRINT "SALDO T
OTAL DE";A$(1);TAB(23);A(1,0)
850 PRINT:PRINT "QUALQUER TECLA
 PARA RECOMEÇAR"
860 IF INKEY$="" THEN 860 ELSE
RUN
900 CLS
905 JK=INT(RND(1)*100)+50:IF JK
>A(M,2) THEN JK=A(M,2)
910 PRINT TAB(9);"R O U B O"
920 PRINT:PRINT "     FORAM ROUB
ADOS";JK;"KG":PRINT "DE SEU OUR
O":A(M,2)=A(M,2)-JK:A(M,0)=A(M,
0)-JK*ER
930 PLAY"T255O1CDEFBAGFED"
940 FOR I=1 TO 500:NEXT:CLS:RET
URN
```

```
20  DIM A(1,5),C(1,4),A$(1)
120  HOME : PRINT "QUANTOS JOGA
DORES (1 OU 2) ?";
130  GET A$: IF A$ < "1" OR A$
> "2" THEN 130
140 P =  VAL (A$):NO = P:ER = 1
0000:A(0,0) = 2000000:A(0,1) =
A(0,0):A(1,0) = A(0,0):A(1,1) =
A(0,0):A(0,3) = 10000:A(1,3) =
A(0,3)
150  FOR N = 1 TO P: PRINT : PR
INT : PRINT "NOME DO JOGADOR ";
N;: INPUT A$(N - 1): IF  LEN (A
$(N - 1)) > 8 THEN A$(N - 1) =
LEFT$ (A$(N - 1),8)
160  NEXT
200  FOR N = 0 TO 29: FOR M = 0
 TO NO - 1
202  HOME
210  INVERSE : HTAB 12: PRINT "
MINA DE OURO ": NORMAL
220  PRINT : PRINT  TAB( 20);A$
(0);: IF NO = 2 THEN  PRINT  TA
B( 30);A$(1);
230  PRINT : PRINT "SALDO TOTAL
"; TAB( 20);A(0,0);: IF NO = 2
THEN  PRINT  TAB( 30);A(1,0);
240  PRINT : PRINT "DINHEIRO";
TAB( 20);A(0,1);: IF NO = 2 THE
N  PRINT  TAB( 30);A(1,1);
250  PRINT : PRINT "OURO    KG";
 TAB( 20),A(0,2);: IF NO = 2 TH
EN  PRINT  TAB( 30);A(1,2);
260  PRINT : PRINT "CUSTO P/ CA
VAR  $"; TAB( 20);A(0,3);: IF N
O = 2 THEN  PRINT  TAB( 30);A(1
,3);
270  PRINT : PRINT "N.O DE MINA
S"; TAB( 20);A(0,4);: IF NO = 2
 THEN  PRINT  TAB( 30);A(1,4);
```



Como assegurar bons resultados na procura de ouro?

Quando o programa estiver completo, você verá que ele segue as regras do comércio — atividade que, entre outras coisas, inclui sorte. Por isso mesmo, é difícil dar a receita do sucesso e da fortuna. Mas algumas dicas terão utilidade.

Embora os custos de mineração possam ser reduzidos graças a pesquisa e desenvolvimento, não é aconselhável investir grandes somas neste item, já que você conta só com trinta rodadas para enriquecer.

Faça escavações somente em minas com boa possibilidade de produzir ouro, a uma pequena profundidade. Não fique, porém, eternamente passando a vez, pois o jogo pode acabar antes que a mina ideal apareça.

Se estocar ouro, você correrá o risco de ser roubado, mas compare as chances de que isso ocorra com a cotação do ouro vigente no mercado — tente sempre vender o metal em períodos de alta.

```
280  PRINT : PRINT "PROFUNDIDAD
E   M", TAB( 20);A(0,5);: IF NO
 = 2 THEN  PRINT  TAB( 30);A(1,
5),
300  PRINT : PRINT : PRINT "COT
ACAO DO OURO :": PRINT ER;" POR
 KG DE OURO"
400  PRINT : PRINT  TAB( 14);A$
(M)
500  PRINT : PRINT "1- PESQUISA
 E DESENVOLVIMENTO"
510  PRINT "2- LEVANTAMENTO GEO
LOGICO"
520  PRINT "3- CAVAR MAIS 200 M
"
530  PRINT "4- VENDER OURO NO M
ERCADO"
540  PRINT "5- PASSAR A VEZ"
600  GET A$: IF A$ < "1" OR A$
> "5" THEN 600
620  ON  VAL (A$) GOSUB 1000,20
00,3000,4000,5000
700  IF A(M,1) < 0 THEN 7000
710 ER = ER +  INT ( RND (1) *
1000) - 500
720  IF  INT ( RND (1) * 1600)
 A(M,2) < 0 THEN  GOSUB 900
740 A(M,0) = A(M,1) + A(M,2) *
ER
750  HOME
790  NEXT M,N
810  HOME
820  INVERSE : HTAB 14: PRINT "
 FIM DE JOGO ": NORMAL
830  VTAB 5: PRINT "SALDO TOTAL
 DE ";A$(0);" : ";A(0,0)
840  IF NO = 2 THEN  PRINT   PR
INT "SALDO TOTAL DE ";A$(1);" :
 ";A(1,0)
850  PRINT : PRINT : PRINT "QUA
LQUER TECLA PARA RECOMECAR"
860  GET A$: RUN
900  HOME
905 JK =  INT ( RND (1) * 100)
+ 49: IF JK > A(M,2) THEN JK =
A(M,2)
910  INVERSE : HTAB 15: PRINT "
 R O U B O "  NORMAL
920  PRINT : PRINT "FORAM ROUBA
DOS ";JK;" KG DE SEU OURO":A(M,
2) = A(M,2) - JK:A(M,0) = A(M,0
) - JK * ER
930  FOR I = 1 TO 2000: NEXT
940  HOME : RETURN
```

T

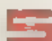
```
10 PMODE 3,1:CLS
20 DIM H(23),T(0),D(1),B(1),A(1
,5),C(1,4),A$(1),R(1)
40 FOR K=0 TO 9:READ NU$(K):NEX
T
50 DATA NR2D4R2U4BR2,BFEND4BR2,
R2D2L2D2R2BU4BR2,NR2BD2NR2BD2R2
U4BR2,D2R2D2U4BR2,NR2D2R2D2L2BE
4,D4R2U2L2BE2BR2,R2ND4BR2,NR2D4
R2U2NL2U2BR2,NR2D2R2D2U4BR2
60 PCLS 3
70 DRAW"BM36,23C2L35U6E3R3NU4R5
U10E2RE3R3F4D3F4DF2DF2DF3D2":PA
INT(18,16),2
80 DRAW"BM24,9C3G2D6F5R3E2UH2UH
UH2UH2BM20,1NLDL2GR5D5BM14,6RBR
3RBD4DBL4UBD3LBR4RBD2LBL3LBD2RB
R3RBD2LBL3LBD2RBR3RBD2LBL3L":PA
INT(26,15),3
90 DRAW"BM2,21C4UBR4ND3BR4D":GE
T(0,0)-(37,23),H,G
100 PCLS:DRAW"BM7,0C4L6BD2ERFRE
RBD2L7DR7DL5GNR3DNR3FNR4DNR4GNR
3DNR3FNR4DR2GL3FNR6FR3FL4GNRDR5
DL3"
110 GET(0,0)-(7,2),T,G:GET(0,3)
-(7,10),D,G:GET(0,11)-(7,20),B,
G
120 PRINT @129,"QUANTOS JOGADOR
ES (1 OU 2) ?";
130 A$=INKEY$:IF A$<"1" OR A$>"
2" THEN 130
140 P=VAL(A$):NO=P:ER=10000:A(0
,0)=2000000:A(0,1)=2000000:A(1,
0)=2000000:A(1,1)=2000000:A(0,3
)=100000:A(1,3)=100000
150 FOR N=1 TO P:PRINT:PRINT:PR
INT" NOME DO JOGADOR";N;:LINEIN
PUT A$(N-1):IF LEN(A$(N-1))>8 T
HEN A$(N-1)=LEFT$(A$(N-1),8)
160 NEXT
200 FOR N=0 TO 29:FOR M=0 TO NO
-1
202 CLS
210 PRINT @3,"mina de ouro",
220 PRINT TAB(15);A$(0);:IF NO=
2 THEN PRINT TAB(24);A$(1)
230 PRINT @32,"SALDO TOTAL";TAB
(14);A(0,0):IF NO=2 THEN PRINT
@55,A(1,0)
240 PRINT @64,"DINHEIRO";TAB(14
),A(0,1):IF NO=2 THEN PRINT @87
,A(1,1)
250 PRINT @96,"OURO kg";TAB(14)
;A(0,2):IF NO=2 THEN PRINT @119
,A(1,2)
260 PRINT @128,"$ P/ CAVAR";TAB
(14);A(0,3):IF NO=2 THEN PRINT
@151,A(1,3)
270 PRINT @160,"NO. DE MINAS";T
AB(14);A(0,4):IF NO=2 THEN PRIN
T @183,A(1,4)
280 PRINT @192,"PROFUNDI// m";T
AB(14);A(0,5):IF NO=2 THEN PRIN
T @215,A(1,5)
300 PRINT @224,"COTACAO DO OURO
 NO MERCADO":PRINT ER,"POR KG D
E OURO "
400 PRINT @330,A$(M)
500 PRINT "1- PESQUISA E DESENV
OLVIMENTO
510 PRINT "2- LEVANTAMNTO GEOLO
GICO"
520 PRINT "3- CAVAR MAIS 200 M"
530 PRINT "4- VENDER OURO NO ME
RCADO"
540 PRINT "5- PASSA A VEZ";
600 A$=INKEY$:IF A$<"1" OR A$>"
5" THEN 600
620 ON VAL(A$) GOSUB 1000,2000,
3000,4000,5000
700 IF A(M,1)<0 THEN 7000
710 ER=ER+RND(1000)-200
720 IF RND(1600)-A(M,2)<0 GOSUB
 900
740 A(M,0)=A(M,1)+A(M,2)*ER
```



```
750 CLS
790 NEXT M,N
810 CLS
820 PRINT @138,"FIM DE JOGO"
830 PRINT @197,"SALDO TOTAL DE"
;A$(0):PRINT TAB(11);A(0,0)
840 IF NO=2 THEN PRINT TAB(5),"
SALDO TOTAL DE",A$(1):PRINT TAB
(11);A(1,0)
850 PRINT @449,"QUALQUER TECLA
PARA RECOMECAR"
860 IF INKEY$="" THEN 860 ELSE
RUN
900 CLS
905 JK=RND(100)+49:IF JK>A(M,2)
 THEN JK=A(M,2)
910 PRINT @10,"R O U B O"
920 PRINT:PRINT"      FORAM ROUB
ADOS";JK;"KG DE":PRINT"      SEU
```

```
OURO":A(M,2)=A(M,2)-JK:A(M,0)=
A(M,0)-JK*ER
930 PLAY"T4O1CDEFBAGFED"
940 CLS RETURN
```

Todos esses programas funcionam de maneira similar, seguindo a mesma estrutura básica e a mesma numeração. Do início até a linha 200, contudo, há algumas variações.



Para começar, a linha 10 pergunta qual o número de jogadores, e a linha 20 verifica se a resposta se encontra no intervalo adequado. A linha 30 acerta os valores de **p** e **nop** de acordo com o número escolhido.

Algumas matrizes são dimensionadas na linha 40, juntamente com a cotação do ouro no mercado — **er**. A matriz **a** armazena informações sobre o saldo dos jogadores; a matriz **c**, informações sobre as minas; a matriz **a$** contém os nomes dos jogadores e a **r** é usada para indicar o começo das escavações na mina em questão. A linha 50 estabelece os saldos e as condições iniciais das minas dos dois jogadores. O valor zero é colocado em **r(1)** e **r(2)**, indicando que as escavações não começaram na primeira mina. Outros valores são dados a seguir: **a(1,1)** e **a(2,1)** contêm o saldo de cada jogador; **a(1,2)** e **a(2,2)**, o total em dinheiro que cada jogador possui; **a(1,3)** e **a(2,3)** indicam a quantidade de ouro de cada um; **a(1,4)** e **a(2,4)**, os custos de mineração; **a(1,5)** e **a(2,5)** revelam o número de minas de cada um e **a(1,6)** e **a(2,6)**, a profundidade das minas.

A linha 70 permite a entrada do nome dos jogadores.



A linha 20 dimensiona as matrizes. A linha 90 seleciona a tela de textos de 32

colunas com fundo amarelo, caracteres pretos e bordas azuis. As teclas de função são desligadas.

A tela de 32 colunas foi escolhida porque permite o uso de cores e de blocos gráficos, tornando o programa mais interessante.

A linha 100 faz com que os oito primeiros caracteres — códigos 0 a 7 — se tornem blocos azuis (veja artigo da página 261). A mesma linha ainda faz com que os oito caracteres seguintes — códigos 8 a 15 — se tornem blocos de cor vermelha.

A linha 110 determina que os caracteres numéricos sejam escritos em branco sobre fundo vermelho.

Depois desses preparativos iniciais, a linha 120 pergunta qual o número de jogadores. A linha 130 verifica se a resposta obtida é válida — um ou dois jogadores.

Em seguida, a linha 140 acerta o valor das variáveis **P**, **NO** — as duas iguais ao número de jogadores — e **ER** — cotação do ouro. Além disso, ela estabelece os valores iniciais de vários elementos da matriz **A**: **A(0,0)** e **A(0,1)**, saldo total dos jogadores; **A(0,1)** e **A(1,1)**, quantia em dinheiro que cada um possui; **A(0,2)** e **A(1,2)**, quantidade de ouro; **A(0,3)** e **A(1,3)**, custo da mineração.

As linhas 150 e 160 perguntam o nome dos jogadores.

O jogo do MSX, como já dissemos, utiliza a tela de textos de 32 colunas, com cor de fundo amarelo e caracteres pretos. Os caracteres numéricos aparecem em branco sobre vermelho, enquanto a tela é mantida com fundo azul. Tudo isso é obtido por meio do comando **VPOKE**, que já usamos outras vezes (veja artigo da página 261).



Na versão para Apple e o TK-2000, nosso jogo começa dimensionando algumas variáveis na linha 20.

Em seguida, a linha 120 pergunta quantas pessoas vão participar e a linha 130 verifica a validade da resposta. A linha 140 guarda o número de jogadores nas variáveis **P** e **NO**, que servirão de contadores e sinalizadores em vários pontos do programa, permitindo que os dados de cada jogador sejam colocados no local correto. Essa linha ainda acerta o valor da cotação do ouro — **ER** —, bem como de vários elementos da matriz **A**: **A(0,0)** e **A(1,0)**, saldo total dos jogadores; **A(0,1)** e **A(1,1)**, total em dinheiro que cada um possui; o par seguinte corresponde à quantidade de ouro e o último, ao custo da escavação nas minas.

As linhas 150 e 160 perguntam o nome dos jogadores.



A linha 10 seleciona o modo **PMODE 3** e a tela é limpa. A linha 20 dimensiona várias matrizes.

Como parte do jogo acontece na tela gráfica, em certos momentos é preciso usar **DRAW** para escrever texto. As linhas 40 a 110 contêm a rotina para escrever na tela de alta resolução, publicada em INPUT anteriormente (veja artigo da página 232).

As linhas 120 e 130 perguntam o número de jogadores e verificam se a resposta é válida. A linha 140 acerta os valores de **P** e **NO** de acordo com o número escolhido. Em seguida, alguns dos elementos da matriz **A** têm seus valores iniciais estabelecidos, a saber: **A(0,0)** e **A(1,0)** contêm o saldo total de cada jogador; o próximo par de elementos contém os saldos em dinheiro; o seguinte, a quantidade de ouro que cada um possui e o último corresponde ao custo da mineração.

As linhas 150 e 160 pedem os nomes dos jogadores.



Daí em diante os programas são bem parecidos. Todos eles têm dois laços **FOR...NEXT** que começam na linha 200 e terminam nas linhas 790 e 800. Esses laços são responsáveis pela exibição do menu e da situação das companhias de mineração na tela.

A variável **N** (**n**, no caso do Spectrum) conta o número de rodadas já jogadas. A variável **NO** (**nop**, no Spectrum) garante a oferta de trinta rodadas a cada jogador. Observe que, mais adiante na listagem, esta mesma variável fará com que a situação das companhias de mineração seja exibida.

A linha 202 escolhe as cores da tela no Spectrum. Nos demais, ela apenas limpa a tela — note que no MSX isto é feito por uma sub-rotina que enche a tela com o caractere de código 0, cuja cor azul foi definida pela linha 110. A linha 210 imprime o título: MINA DE OURO. A linha 220 imprime o nome dos jogadores. O segundo nome só é impresso se a opção para dois jogadores foi escolhida no início.

As linhas 230 a 300 mostram os valores: SALDO TOTAL, saldo em DINHEIRO, saldo em OURO, custo para CAVAR, Nº DE MINAS, PROFUNDIDADE da mina e COTAÇÃO DO OURO NO MERCADO. Se duas pessoas estiverem jogando, os dois valores serão impressos no local correspondente. O programa faz isso com base nos valores das variáveis **nop** ou **NO**.

A linha 400 mostra o nome da pessoa que está jogando no momento. As linhas 500 a 540 oferecem as opções de Pesquisa e Desenvolvimento, Levantamento Geológico, Cavar mais 200 metros, Vender ouro no mercado ou Passar a vez. No Spectrum, a linha 550 solicita ao jogador que faça sua opção. Nos demais micros não há esta linha.

As linhas 600 a 620 usam **INKEY$** ou **GET** para obter a resposta do jogador. Além disso, verificam a validade da resposta e chamam a sub-rotina correspondente à escolha feita.

A linha 700 verifica se o saldo total caiu abaixo de zero, terminando o programa, se for o caso, por meio de uma sub-rotina de finalização. A linha 710 faz com que a cotação do ouro flutue ao acaso; assim, o jogador deverá ter o cuidado de vender o metal nas épocas de "alta".

A linha 720 compara um número aleatório com a quantidade de ouro do jogador, para decidir se haverá um roubo — observe que as chances de roubo aumentam com a quantidade de ouro armazenada. A rotina que cuida do roubo fica nas linhas 900 a 940. A linha 905 decide quanto ouro vai ser roubado, e a linha 920 informa que esta quantidade foi roubada.

A linha 740 calcula o saldo total, adicionando o saldo em dinheiro ao valor do ouro armazenado, calculado com base na cotação atual.

A linha 350 restabelece as cores da tela e a limpa, antes que o **NEXT** envie o programa de volta para a linha 200, para mais uma rodada.

As linhas 810 a 840 contêm a rotina de finalização do jogo, usada quando o saldo de um dos jogadores cai abaixo de zero. Ela mostra na tela as condições finais das companhias de mineração e informa: FIM DE JOGO.

Finalmente, as linhas 850 a 880 permitem uma nova partida.

Na próxima seção, adicionaremos várias sub-rotinas que farão com que o jogo funcione. Haverá rotinas para reduzir os custos de mineração por meio de pesquisa e desenvolvimento, para fazer o levantamento geológico, escavar o solo e vender o ouro no mercado. Além disso, traremos os dados necessários para a elaboração dos gráficos que ilustram as minas e mostram o progresso das escavações.

SPRITES PARA O TRS-80 (3)

- OS CARACTERES DE CONTROLE DO CURSOR
- FIGURAS MAIS COMPLEXAS
- ANIMAÇÃO DE FIGURAS COMPLEXAS

Usando gráficos de baixa resolução, podemos criar figuras semelhantes a sprites. Você aprenderá a armazená-las em uma única variável, mesmo que ocupem várias linhas na tela.

A forma mais avançada de produção de figuras animadas em BASIC, nos micros da linha TRS-80, consiste na concatenação de vários caracteres gráficos e seu armazenamento em uma única variável alfanumérica.

Esse método funciona muito bem, mesmo quando a figura ocupa mais de uma linha na tela. Mas, para que possamos continuar tratando a figura como uma entidade única (por exemplo, para empregar a variável que a armazena dentro de um **PRINT@**), precisaremos incluir na sua definição *caracteres de controle do cursor*.

Já mencionamos, brevemente, que certos caracteres situados na faixa do código ASCII que vai de 0 a 31 servem para o controle de funções do vídeo, quando são utilizados com um **PRINT CHR$**. Por exemplo, **PRINT CHR$(23)** dobra a largura dos caracteres na tela (32 colunas).

Quatro desses caracteres nos interessam na geração de figuras com a técnica que aqui examinaremos. São os caracteres de controle do cursor:

24 recua o cursor uma posição na horizontal
25 avança o cursor uma posição na horizontal
26 desce o cursor uma posição
27 sobe o cursor uma posição

Suponhamos que você queira desenhar na tela um pequeno disco voador, formado por duas linhas. Os códigos da linha de cima serão:

```
100 DATA   174,187,187,187,187,
    187,132
```

e os da linha de baixo:

```
110 DATA   128,143,143,143,143,
    143
```

Para concatenar os códigos em uma única figura, proceda assim: quando o cursor terminar de traçar o último caractere da linha de cima (o 132), faça-o descer uma linha e recuar sete posições (número de bytes da linha de cima do gráfico). Coloque, então:

```
105 DATA   26,24,24,24,24,24,24,
    24
```

Depois, acione as linhas anteriores ao programa que damos a seguir e execute-o para ver o resultado

```
10 D$=""
20 FOR I=1 TO 21 READ N
30 D$=D$+CHR$(N)·NEXT I
40 CLS·PRINT @ 288,D$;
```

A figura aparecerá instantaneamente na tela, com um único **PRINT**!

COMPACTANDO A DEFINIÇÃO

Como a definição da figura inclui muitos códigos idênticos, é possível dar-lhe uma forma mais compacta, recorrendo à função **STRING$** e evitando o laço **FOR...NEXT** e as linhas **DATA**. Digite **NEW** e entre este programa:

```
10 D$   =     CHR$(174)+STRING$
   (5,187)+CHR$(132)+CHR$(26) +
   STRING$(7,24)+CHR$(128)    +
   STRING$(5,143)
20 CLS:PRINT @256,D$;
```

FIGURAS MAIS COMPLEXAS

O método de concatenação de códigos gráficos em uma variável alfanumérica tem, evidentemente, suas limitações. Se a figura ocupar várias linhas, por exemplo, haverá uma excessiva utilização da memória, já que precisamos de um conjunto de caracteres de controle entre cada linha.

A limitação mais séria, entretanto, está na impossibilidade de se criar uma figura com um número de bytes superior a 255, pois este é o comprimento máximo de uma cadeia de caracteres no BASIC do TRS-80. A melhor alternativa será, então, repartir a figura em vários blocos e usar mais de um **PRINT@** para colocá-la inteira na tela. Ao contrário do que parece, tal procedimento não reduz a velocidade de animação gráfica e os resultados, em geral, são semelhantes aos obtidos com a técnica de movimentação do cursor.

Outra técnica muito útil quando se tem figuras grandes e complexas é a chamada *tríade de posição, comprimento, caractere*. Suponhamos que você queira desenhar um trenzinho, fazendo o pistão da roda se movimentar e nuvens de fumaça escaparem da chaminé. O mais adequado será desenhar primeiro a locomotiva completa, usando a técnica da tríade e, depois, animar as partes que interessam, substituindo sucessivamente duas figuras (por exemplo, o braço do pistão em duas posições).

A tríade consiste em uma sequência de três números, expressa em uma linha **DATA** de definição da figura. O primeiro número determina a posição @ onde começa a cadeia de caracteres; o segundo define a quantidade de caracteres existentes na cadeia; o terceiro indica o código gráfico dos elementos iguais da cadeia:

```
100 FOR I=1 TO 32·READ N1,N2,N3
110 PRINT @ N1,STRING$(N2,N3);
120 NEXT I
```

ACELERE OS SPRITES

Agora que você já sabe como criar figuras de características semelhantes às dos sprites para os microcomputadores da linha TRS-80, será fácil fazer animação gráfica com eles. É claro que essas figuras não possuem uma propriedade fundamental dos sprites autênticos (como os da linha MSX): a *priorização* de imagens. Explicando melhor, cada sprite recebe um número, que indica ao computador se ele vai obstruir ou ser obstruído por outro sprite que ocupar o mesmo lugar na tela.

De qualquer maneira, é possível elaborar programas de animação gráfica relativamente sofisticados para o TRS-80. Um fator importante na animação é o tempo que leva o computador para traçar um bloco gráfico, definido conforme as técnicas apresentadas nesta série. Se a execução do desenho for muito demorada — como no caso de blocos gráficos grandes ou formados por vários **PRINT @** separados —, a animação gráfica não sairá perfeita.

SIMULAÇÃO: FAÇA A BOLA ROLAR

A matemática fornece vários modelos para o mundo real. Utilizando-os em seus programas, você pode produzir efeitos muito interessantes. Aqui, simulamos a trajetória de uma bola.

Na tentativa de construir um modelo para o mundo que nos cerca, a matemática aplicada e outros ramos da ciência fazem uso de um grande número de equações. Estas resultam do exame de dados gerados por observações e experimentos. A partir desse exame, calcula-se uma equação que explica a relação encontrada de maneira que satisfaça os dados obtidos.

Tais equações relacionam-se à mais variada gama de fenômenos físicos e biológicos — desde o crescimento de uma planta até a trajetória de um cometa no espaço. Esse tipo de informação tem diversos usos, sobretudo porque as equações possibilitam prever como as coisas se comportarão em determinadas circunstâncias.

O significado das equações mais complicadas pode não ser claro à primeira vista. Algumas delas só são compreensíveis para os que estão envolvidos num campo específico de pesquisa. Muitas, porém, descrevem fenômenos que fazem parte da experiência de qualquer pessoa. Já vimos um exemplo no programa que usa uma fórmula para estudar o comportamento de um objeto em queda (veja artigo da página 434).

Mas, sejam as equações complicadas ou não, o resultado dos cálculos consiste simplesmente num número — para o tempo ou qualquer outra coisa —, que é mais um modelo para o mundo real. Além da utilidade já mencionada, essas equações têm outro emprego: elas podem ser programadas no seu micro para produzir uma interessante simulação da realidade. Nesse caso, a equação é utilizada para controlar a apresentação de tela. Assim, no exemplo anterior, poderíamos criar uma imagem para mostrar como se comporta um objeto em queda livre.

É possível empregar qualquer equação de movimento para imitar na tela a movimentação real de um objeto. Essa idéia está na base de muitos programas e de curiosas animações.

A melhor maneira de compreender como isso funciona é trabalhar com um exemplo prático — ou seja, tomar uma série de equações que podem ser encontradas em qualquer livro de física e transformá-las em belas simulações na tela. Como exemplo, vamos começar com algo fácil de simular: a trajetória de uma bola numa mesa de bilhar.

O MODELO DA BOLA

Se você bater numa bola com um taco (ou lhe der um chute), ela se moverá em linha reta até encontrar algum obstáculo. Se houver outra força agindo sobre ela — ventos laterais ou gravidade, digamos —, a trajetória e a velocidade serão modificadas. Em uma boa mesa de bilhar, pode-se desconsiderar a ação de ventos, restando apenas uma força com que se preocupar: o efeito da fricção ou atrito com a mesa, que reduzirá gradualmente a velocidade da bola. Mas, quanto? Aí entra a primeira de nossas equações.

Grosso modo, quanto maior a força que atingir a bola no momento da tacada, mais longe ela irá. Existe uma



- O MODELO DA BOLA
- COMO FAZER A BOLA ROLAR
- UMA MIRA MÓVEL
- COMO A EQUAÇÃO FUNCIONA
- TACADAS MAIS FORTES
- CÁLCULO DAS COORDENADAS
- RAÍZES QUADRADAS
- COMO DESACELERAR
- COMO CALCULAR A VELOCIDADE DA BOLA

equação que calcula a distância a ser percorrida por um objeto, dada a força inicial e alguns outros elementos, como o efeito do atrito. Essa equação tem variadas aplicações em programação. Para vê-la funcionando, digite e execute este programa:

```
10 CLS
20 PRINT "1) Bola de futebol
/ grama longa"
30 PRINT '"2) Bola de golfe /
grama curta"
40 PRINT '"3) Bola de golfe n
a Lua"
50 PRINT '"4) Bola em mesa de
bilhar"
60 INPUT "Faca a sua opcao",o
pt
70 IF opt<1 OR opt>4 THEN
GOTO 60
80 IF opt=1 THEN  LET fg=4
90 IF opt=2 THEN  LET fg=1.4
95 IF opt=3 THEN  LET fg=.1
100 IF opt=4 THEN  LET fg=.7
110 CLS
115 PRINT "A velocidade pode s
er de 1 a 15 metros por segund
o"
120 INPUT "Qual a velocidade i
nicial (m/s)",v
125 IF v<1 OR v>15 THEN  GOTO
 120
160 LET dist=(v*v)/(2*fg)
170 CLS
210 PRINT AT 2,4;"Sua bola alc
ancaria"
220 PRINT AT 7,8;dist;"metros"
250 PRINT AT 18,1; FLASH 1;"Qu
alquer tecla para continuar"
260 PAUSE 0
270 GOTO 10
```





```
10 CLS
20 PRINT @96," 1)BOLA DE FUTEBO
L / GRAMA LONGA"
30 PRINT @160," 2)BOLA DE FUTEB
OL / GRAMA CURTA"
40 PRINT @224," 3)BOLA DE GOLFE
 NA LUA"
50 PRINT @288," 4)BOLA EM MESA
DE BILHAR"
60 PRINT:PRINT" QUAL SUA OPCAO
?"
70 K$=INKEY$:IF K$<"1" OR K$>"4
" THEN 70
80 OP=VAL(K$)
90 IF OP=1 THEN FG=4
100 IF OP=2 THEN FG=1.4
110 IF OP=3 THEN FG=.1
120 IF OP=4 THEN FG=.7
130 CLS
140 PRINT" A VELOCIDADE PODE SE
R DE 1 A 15 METROS POR SEGUNDO"
150 PRINT:INPUT" QUAL A VELOCID
ADE INICIAL QUE   VOCE DESEJA D
AR (M/S)";VE
160 IF VE<1 OR VE>15 THEN 130
170 DI=(VE*VE)/(2*FG)
180 CLS
190 PRINT"SUA BOLA ALCANCARIA",
DI;"METROS"
200 PRINT @449,"QUALQUER TECLA
PARA RECOMECAR"
210 IF INKEY$="" THEN 210
220 GOTO 10
```

Para rodar o programa no TRS-80, modifique os números utilizados nas instruções **PRINT@** para o dobro dos que estão na listagem.



```
10 CLS:COLOR 15,4,4:DEFSNG D
20 LOCATE 5,5:PRINT"1) Bola de
futebol em grama longa"
```

```
30 LOCATE 5,7:PRINT"2) Bola de
golfe em grama curta"
40 LOCATE 5,9:PRINT"3) Bola de
golfe na lua"
50 LOCATE 5,11:PRINT"4) Bola em
 mesa de bilhar"
60 LOCATE 7,18:PRINT"Escolha su
a opção:";
70 K$=INKEY$:IF K$<"1" OR K$>"4
" THEN 70
80 OP=VAL(K$)
90 IF OP=1 THEN FG=4
100 IF OP=2 THEN FG=1.4
110 IF OP=3 THEN FG=.1
120 IF OP=4 THEN FG=.7
130 CLS
140 PRINT:PRINT"  A velocidade
em metros por segundo  pode ser
 de 1 a 15."
150 PRINT:INPUT"  Qual velocida
de inicial (em m/s)";VE
160 IF VE<1 OR VE>15 THEN 130
170 DI=(VE*VE)/(2*FG)
180 CLS
190 PRINT:PRINT"  Sua bola perc
orreu ";DI;"metros"
200 LOCATE 0,12:PRINT"Pressione
 qualquer tecla para recomeçar"
210 IF INKEY$="" THEN 210
220 GOTO 10
```



```
10  HOME
20  HTAB 5: VTAB 5: PRINT "1) B
OLA DE FUTEBOL EM GRAMA LONGA"
30  HTAB 5: VTAB 7: PRINT "2) B
OLA DE GOLFE EM GRAMA CURTA"
40  HTAB 5: VTAB 9: PRINT "3) B
OLA DE GOLFE NA LUA"
50  HTAB 5: VTAB 11: PRINT "4)
BOLA EM MESA DE BILHAR"
60  HTAB 8: VTAB 15: PRINT "ESC
OLHA SUA OPCAO ";
70  GET K$: IF K$ < "1" OR K$ >
```

```
"4" THEN 70
80 OP =  VAL (K$)
90  IF OP = 1 THEN FG = 4
100  IF OP = 2 THEN FG = 1.4
110  IF OP = 3 THEN FG = .1
120  IF OP = 4 THEN FG = .7
130  HOME
140  PRINT : PRINT "  A VELOCID
ADE EM METROS POR SEGUNDO PO-DE
  SER DE 1 A 15."
150  PRINT : INPUT "  QUAL A VE
LOCIDADE INICIAL (EM M/S) ";VE
160  IF VE < 1 OR VE > 15 THEN
130
170 DI = (VE * VE) / (2 * FG)
180  HOME
190  PRINT : PRINT "  SUA BOLA
PERCORREU ",DI;" METROS"
200  VTAB 12  PRINT "PRESSIONE
QUALQUER TECLA PARA RECOMECAR";

210  GET K$
220  GOTO 10
```

O programa começa perguntando a respeito das condições em que se encontra nossa bola hipotética. Em seguida, calcula a distância que ela percorreria, para qualquer velocidade inicial que você escolher. Vários fatores — como atrito, massa e gravidade — são levados em conta.

A equação usada é a seguinte:

$$\text{Distância} = \frac{v\uparrow 2}{2*fg}$$

O v corresponde à velocidade inicial e **fg** ao valor que engloba a gravidade e o atrito. Obviamente, o valor de **fg** é diferente para cada uma das opções do programa.

OS RESULTADOS

A equação é solucionada na linha 160 do programa do Spectrum ou 170 do dos outros micros. Você pode ver que, embora na equação haja uma potenciação, não há sinal dessa operação (↑) na linha referida. Ocorre que os computadores calculam mais rapidamente somas e multiplicações do que potências. Assim, quando o número está elevado ao quadrado (nosso caso), é melhor multiplicá-lo por ele mesmo (v*v) do que elevá-lo à segunda potência (v ↑ 2).

A opção 3 — bola de golfe na Lua — fornece resultados muito maiores que as demais porque o seu valor para **fg** é bem menor que o das outras opções, uma vez que a gravidade na Lua corresponde a somente um sexto da gravidade na Terra.

A última opção — a da bola na mesa de bilhar, com a qual trabalharemos a seguir — também oferece resultados altos, porque o atrito é muito pequeno em relação ao da grama.

Os valores do programa para o atrito e a gravidade são apenas aproximados, não exatos. Porém, dão uma idéia razoável de como uma equação desse tipo pode ser usada dentro de um programa para prever a distância percorrida por um objeto sob diversas circunstâncias. Mas ainda não pudemos visualizar o objeto em movimento.

A BOLA EM AÇÃO

A equação apresentada calcula a distância percorrida por um objeto, mas assume que não há nada no caminho dele. A opção da bola de bilhar, no programa dado, fornece um bom exemplo: o resultado pode ser uma distância de 10 metros. Mas mesas de bilhar não têm esse tamanho: a bola rebateria nas bordas da mesa.

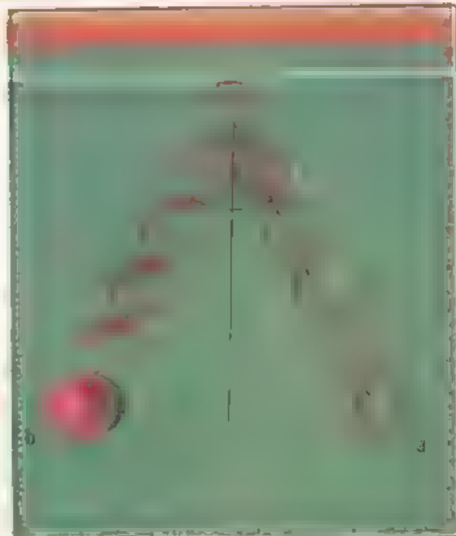
Ao contrário do comportamento imprevisível de uma bola de futebol sobre a grama, uma bola de bilhar reage de maneira muito regular quando bate contra a borda da mesa (a não ser que tenha tomado um efeito giratório). Ela rebate espalhando o ângulo de incidência (o ângulo com que a bola atinge a borda). Observando a figura da página 673, você entenderá melhor o que ocorre. A bola saiu do ponto **a** e moveu-se pela reta tracejada até parar no ponto **b**. Como você pode ver, os ângulos formados pela linha da trajetória da bola e uma reta perpendicular à borda da mesa são simétricos.

Mais uma vez, a matemática aplicada fornece uma equação para demonstrar este princípio. Com isso, podemos fazer um programa que simule a trajetória da bola na mesa rebatendo em uma ou mais bordas.

O quadro da última página deste artigo mostra as equações que calculam as posições da bola após bater contra qualquer uma das bordas da mesa. Embora pareça complicado, seu computador faz os cálculos muito rapidamente.

S

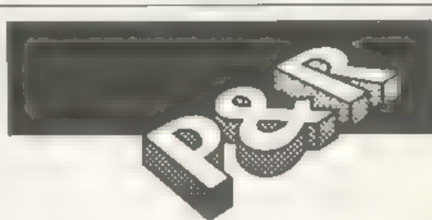
```
10 BORDER 4: PAPER 4· INK 0:
CLS
60 LET bx=128· LET by=76: LET
cx=100: LET cy=112: LET nx=
100: LET ny=112
90 FOR n=6 TO 20: PRINT
PAPER 7;AT n,2;"
                    ": NEXT n
100 CIRCLE  OVER 1;bx,by,4
110 PLOT  OVER 1;cx-4,cy: DRAW
 OVER 1;8,0: PLOT  OVER 1;cx,
cy-4: DRAW  OVER 1;0,8
```



Os ângulos formados pela trajetória da bola de bilhar e uma reta perpendicular à borda da mesa são simétricos.

```
120 LET a$=INKEY$: IF a$=""
THEN  GOTO 120
130 IF INKEY$="a" AND cy>12
THEN  LET ny=cy-2· GOTO 190
140 IF INKEY$="q" AND cy<122
THEN  LET ny=cy+2: GOTO 190
150 IF INKEY$="p" AND cx<236
THEN  LET nx=cx+2: GOTO 190
160 IF INKEY$="o" AND cx>20
THEN  LET nx=cx-2: GOTO 190
170 IF INKEY$=CHR$ 32 THEN
GOSUB 500: GOTO 300
190 PLOT  OVER 1;cx-4,cy: DRAW
 OVER 1;8,0: PLOT  OVER 1;cx,
cy-4: DRAW  OVER 1;0,8
200 LET cy=ny: LET cx=nx
210 GOTO 110
300 IF INKEY$="" THEN  GOTO
 300
305 CLS
310 PRINT AT 2,5;"ALCANCE=";
INT (p*p*100/(18*18*.4))/100
320 PRINT AT 5,5;"VELOCIDADE I
NICIAL=";INT (ABS (p*100/18))/
100
330 PRINT AT 10,8;"Outra vez ?
(s/n)"
340 LET a$=INKEY$: IF a$<>"s"
AND a$<>"n" THEN  GOTO 340
350 IF a$="s" THEN  CLS : GOTO
90
360 PAPER 7: CLS · STOP
500 IF cx=bx AND cy=by THEN
LET p=0: RETURN
510 LET p=0
515 PRINT AT 0,0;"
                      "
520 LET p=p+2: IF p=256 THEN
GOTO 510
530 PLOT p-1,168: DRAW 0,7:
PLOT p,168: DRAW 0,7
540 IF INKEY$=CHR$ 32 THEN
GOTO 520
550 CIRCLE  OVER 1;bx,by,4
560 LET dx=cx-bx: LET dy=cy-by
570 LET v=p/18: LET sq=SQR (dx
```

```
*dx+dy*dy)
590 LET t=1
600 PLOT bx=.5,by+.5
610 LET vx=v*dx/sq: LET vy=v*
dy/sq
620 LET bx=bx+vx*t-SGN vx*.1*t
*t: LET by=by+vy*t-SGN vy*.1*t
*t
630 LET v=v-.1
640 IF ABS v<.1 THEN  GOTO 700
650 IF bx<15 THEN  LET dx=-dx:
LET bx=30-bx
660 IF bx>239 THEN  LET dx=-dx
: LET bx=478-bx
670 IF by<8 THEN  LET dy=-dy:
```



Como variar a velocidade de simulação do movimento?

Em primeiro lugar, é necessário estabelecer a distinção entre velocidade simulada de deslocamento da bola (ou de um corpo qualquer), medida em cm/s ou outra unidade de velocidade, e velocidade real de deslocamento da imagem da bola na tela.

A velocidade real com que a bola se deslocará na tela depende de muitos fatores. Um deles é o tempo que o interpretador BASIC leva para resolver todas as equações matemáticas que regem o modelo cinemático. Esse tempo varia conforme o computador e a eficiência do programa. Um micro da linha MSX, por exemplo, é bem mais rápido para resolver equações do que um ZX-81. Este divide o tempo total de execução entre UCP e manutenção da tela, o que retarda o cálculo, pois torna necessário o uso de **SLOW** para que a trajetória da bola seja exibida na tela.

A eficiência do programa também é importante. Um número excessivo de substituições nas equações retarda a solução. Podemos, assim, aumentar a eficiência, usando alguns truques, como a inclusão de poucas equações ou a colocação de várias delas em uma única linha de comando.

É importante, ainda, considerar que o incremento de distância nas direções X e Y (**DX** e **DY**) afeta o número de pontos de trajetória calculados por segundo. Quanto menores esses valores, mais fiel será a simulação e mais precisos serão os resultados dos cálculos. Infelizmente, a velocidade diminuirá. Usar valores muito altos de **DX** e **DY**, por outro lado, é perigoso, pois a simulação pode ficar pouco realista.

```
LET by=16-by
680 IF by>127 THEN  LET dy=-dy
: LET by=254-by
690 GOTO 600
710 IF bx<19 THEN  LET bx=20
720 IF bx<235 THEN  LET bx=235
730 IF by<12 THEN  LET by=12
740 IF by>123 THEN  LET by=123
750 CIRCLE bx,by,4: RETURN
```



```
10 PMODE 3,1:PCLS:DIM B(2),BL(2
),C(0)
30 DRAW"BM5,0C3FRFD6GLGHL2HU6E"
:PAINT(5,5),3
40 DRAW"BM20,0C2D2NLR2DL3FD"
50 GET(0,0)-(9,10),B,G
60 GET(18,0)-(23,5),C,G
70 PCLS 4:SCREEN 1,0
80 LINE(18,58)-(230,170),PRESET
,BF
90 BX=123:BY=110:CX=125:CY=112:
NX=125:NY=112
100 PUT(BX,BY)-(BX+9,BY+10),B,O
R
110 PUT(CX,CY)-(CX+5,CY+5),C,OR
120 NX=CX:NY=CY
130 IF PEEK(341)=247 THEN NY=NY
-2:GOTO 190
140 IF PEEK(342)=247 THEN NY=NY
+2:GOTO 190
150 IF PEEK(343)=247 THEN NX=NX
-2:GOTO 190
160 IF PEEK(344)=247 THEN NX=NX
+2:GOTO 190
170 IF PEEK(345)=247 GOSUB 500·
GOTO 220
180 GOTO 100
190 IF NX<18 OR NX>225 OR NY<58
 OR NY>165 THEN 120
200 PUT(CX,CY)-(CX+5,CY+5),BL,P
SET
210 CX=NX:CY=NY:GOTO 100
220 PUT(BX,BY)-(BX+9,BY+10),B,O
R·PUT(CX,CY)-(CX+5,CY+5),C,OR
230 A$=INKEY$
240 IF INKEY$="" THEN 240
250 CLS:PRINT @33,"ALCANCE=";IN
T(P*P*100/(18*18*.4))/100
260 PRINT @97,"VELOCIDADE INICI
AL=";INT(ABS(P*100/18))/100
270 PRINT @225,"QUER RECOMECAR
(S/N) ?"
280 A$=INKEY$:IF A$<>"S" AND A$
<>"N" THEN 280
290 IF A$="S" THEN SCREEN 1,0:G
OTO 100
300 CLS:END
500 IF CX=BX+2 AND CY=BY+2 THEN
 RETURN
510 P=1
520 P=255 AND (P+2):POKE 345,25
5
530 LINE(P,0)-(P+2,6),PSET,BF:L
INE(P+4,0)-(255,6),PRESET,BF
540 SOUND P,1:IF PEEK(345)=247
THEN 520
550 LINE(18,58)-(230,170),PRESE
T,BF
560 DX=CX-BX-2·DY=CY-3-BY
570 V=P/18:SQ=SQR(DX*DX+DY*DY)
580 BX=BX+4:BY=BY+5:T=1
600 PSET(BX+.5,BY+.5,3)
610 VX=V*DX/SQ:VY=V*DY/SQ
620 BX=BX+VX*T-SGN(VX)*.1*T*T:B
Y=BY+VY*T-SGN(VY)*.1*T*T
630 V=V-.1:IF ABS(V)<.1 THEN 70
0
650 IF BX<18 THEN DX=-DX:BX=36-
BX:SOUND 5,1
660 IF BX>230 THEN DX=-DX:BX=46
0-BX:SOUND 5,1
670 IF BY<58 THEN DY=-DY:BY=116
-BY:SOUND 5,1
680 IF BY>170 THEN DY=-DY:BY=34
0-BY:SOUND 5,1
690 GOTO 600
700 BX=BX-4:BY=BY-5
710 IF BX<18 THEN BX=18
720 IF BX>221 THEN BX=221
730 IF BY<58 THEN BY=58
740 IF BY>160 THEN BY=160
750 RETURN
```



```
10 CLS·COLOR 15,15,15·SCREEN 2,
0
15 BX=123:BY=100·CX=123:CY=100
NX=CX·NY=CY
20 RESTORE FOR I=1 TO 8
30 READ A·S1$=S1$+CHR$(A)·NEXT
40 FOR I=1 TO 8
50 READ B:S2$=S2$+CHR$(B):NEXT
60 SPRITE$(0)=S1$
70 SPRITE$(1)=S2$
80 LINE(18,38)-(230,170),12,BF
100 PUTSPRITE0,(BX,BY),8
110 PUTSPRITE1,(CX,CY),15
120 K$=INKEY$ IF K$=""THEN 120
130 IF K$=CHR$(28) THEN NX=NX+2
 GOTO 190
140 IF K$=CHR$(29) THEN NX=NX 2
 GOTO 190
150 IF K$=CHR$(30) THEN NY=NY 2
·GOTO 190
160 IF K$=CHR$(31) THEN NY=NY+2
 GOTO 190
170 IF K$=CHR$(32) THEN GOSUB 5
00 GOTO 220
180 GOTO 120
190 IF NX<18OR NX>225 OR NY<38
OR NY>165 THEN NX=CX·NY=CY GOTO
 120
200 CX=NX CY=NY GOTO 100
220 K$=INKEY$ IF K$=""THEN220
230 COLOR 15,4,4:SCREEN0:LOCATE
 5,5 PRINT"Distância percorrida
 ",INT(P*P*100/(18*18* 4))/100
240 LOCATE 5,8·PRINT"Velocidade
 inicial  ";INT(ABS(P*100/18))/
100
250 LOCATE 5,11 PRINT"Outra vez
? (S/N)"
260 K$=INKEY$ IF K$<>"S" AND K$
<>"N" THEN 260
270 IF K$="S" THEN COLOR15,15,1
5 SCREEN 2,0·GOTO 20  ELSE CLS·
END
500 IF CX=BX AND CY=BY THEN RET
URN
510 P=1
520 P=255AND(P+2)
530 LINE(P,0)-(P+2,6),6,BF·LINE
```



```
P+4 0  ,55 6),15,BF
540 IF INKEY$<>CHR$(32) THEN 52
0
560 DX=CX BX DY CY BY
570 V=P/18 SQ=SQR(DX*DX+DY*DY)
580 T 1
600 PSET(BX+4 5,BY+5 5),15
605 PUTSPRITE0,(BX,BY),8
    VX=V*DX/SQ VY=V*DY/SQ
620 BX=BX+VX*T-SGN(VX)*.1*T*T:B
Y=BY+VY*T-SGN(VY)* 1*T*T
630 V=V-.1:IF ABS(V)< 1THEN700
650 IFBX<18THENDX=-DX BX=42-BX
660 IFBX>222THENDX=-DX:BX=444-B
X
670 IFBY<38THENDY=-DY BY=76-BY
680 IFBY>162THENDY=-DY:BY=324-B
Y
690 GOTO 600
700 RETURN
1000 DATA 0,28,62,127,127,127,6
2,28
1010 DATA 0,0,8,8,62,8,8,0
```

```
10  HOME · HGR · HCOLOR= 1· HPL
OT 0,0· CALL 62454
30  FOR I = 1 TO 65
40  READ A
50  POKE 767 + I,A
60  NEXT   POKE 232,00· POKE 23
3,03
80 BX = 123·BY = 110:CX = BX:CY
 = BY·NX = CX NY = CY·P = 1
90  HCOLOR= 0: FOR I = 30 TO 15
0  HPLOT 30,I TO 250,I· NEXT :
HCOLOR= 3
100  SCALE= 1: ROT= 0: DRAW 2 A
T CX + 4,CY + 4
110  DRAW 1 AT BX,BY
120  GET A$
130  IF A$ = "K" THEN NX = NX +
 2: GOTO 190
140  IF A$ = "J" THEN NX = NX -
 2· GOTO 190
150  IF A$ = "I" THEN NY = NY -
 2· GOTO 190
160  IF A$ = "M" THEN NY = NY +
 2  GOTO 190
170  IF A$ =  CHR$ (32) THEN  G
OSUB 500  GOTO 220
180  GOTO 120
190  IF NX < 30 OR NX > 250 OR
NY < 30 OR NY > 150 THEN NX = C
X:NY = CY: GOTO 120
200  GOSUB 750
210 CX = NX·CY = NY· GOTO 100
220  VTAB 21· PRINT "DISTANCIA
PERCORRIDA: "; INT (P * P * 100
 / (18 * 18 *  4)) / 100
230  PRINT "VELOCIDADE INICIAL
 "; INT ( ABS (P * 100 / 18)) /
 100
240  PRINT · PRINT "MAIS UMA VE
Z? ";
250  GET A$  IF A$ <  > "S" AND
 A$ <  > "N" THEN 250
260  IF A$ = "S" THEN  HOME . G
OTO 90
270  TEXT   HOME   END
500  IF CX = BX AND CY = BY THE
N  RETURN
510 K = 2: HCOLOR= K
520 P = P + K  IF P = 255 THEN
K =  - 2: HCOLOR= 1
525  IF P = 1 THEN 510
530  HPLOT P - 1,0 TO P - 1,8:
HPLOT P,0 TO P,8
540  GET A$: IF A$ =  CHR$ (32)
 THEN 520
560 DX = CX - BX·DY = CY - BY
570 V = P / 18·SQ =  SQR (DX *
DX + DY * DY)
580 T = 1
600  DRAW 2 AT CX + 4,CY + 4· D
RAW 1 AT BX,BY
610 VX = V * DX / SQ·VY = V * D
Y / SQ
615  HPLOT BX + 4 - ( SGN (VX)
* 6),BY + 4 - ( SGN (VY) * 6)
620 MX = BX + VX * T -  SGN (VX
) *  1 * T * T:MY = BY + VY * T
 -  SGN (VY) * .1 * T * T:
630 V = V     1: IF  ABS (V) < .
```

```
1 THEN 700
650  IF MX < 30 THEN DX =  - DX
:MX = 64 - MX
660  IF MX > 240 THEN DX =  - D
X·MX = 480 - MX
670  IF MY < 30 THEN DY =  - DY
·MY = 64 - MY
680  IF MY > 140 THEN DY =  - D
Y:MY = 280   MY
690  GOSUB 750.BX = MX:BY = MY·
 GOTO 600
700  RETURN
750  HCOLOR= 0· DRAW 1 AT BX,BY
: DRAW 2 AT CX + 4,CY + 4: HCOL
OR= 3: RETURN
1000  DATA  2,0,6,0,45,0,9,45,4
5,21,63,63,63,23,45,45,45,45,62
,63,63,63,55,45,45,45,45,62,63,
63,63,55,45,45,45,45,30,63,63,6
3,14,45,45,150,0
1010  DATA  36,36,55,54,62,63,5
5,45,45,45,45,37,63,191,50,54,3
9,36,4,0
```

Para o TK-2000, faça as seguintes modificações no programa anterior:

```
10  HOME · HGR · HCOLOR= 1
15  FOR I = 1 TO 160: HPLOT 1,I
 TO 279,I: NEXT
```

O programa desenha um retângulo no meio da tela: esta é a área (ou a mesa) na qual o objeto pode se mover.

UMA MIRA MÓVEL

No meio do retângulo há uma bola e, também, um cursor que pode estar escondido atrás dela. No Spectrum, pressione a tecla Q para movê-lo para cima, A para baixo, O para a esquerda e P para a direita; no TRS-Color e no MSX, use as setas para a movimentação; no Apple e no TK-2000, tecle I para cima, M para baixo, J para a esquerda e K para a direita. O cursor é representado por uma cruz, que se move por toda a área disponível para indicar a direção de movimento da bola. Quando tiver posicionando o cursor no lugar desejado, pressione a barra de espaços (ou a tecla **SPACE**, no Spectrum).

Uma linha horizontal surgirá no topo da tela, aumentando à medida que se mantém a barra de espaços apertada. O tamanho da linha mostra a força com que a bola será impulsionada: quanto maior a linha, maior a força.

Assim que se solta a barra de espaços, o computador limpa o retângulo e desenha uma série de pontos que mostram a trajetória da bola. Os pontos ficam mais próximos quando a bola perde velocidade. No Apple, para colocar a bola em movimento, é necessário pressionar uma tecla qualquer após ter definido a força.

Tente variar a posição do cursor e usar diferentes velocidades na bola, para verificar o efeito.

COMO FUNCIONA

Aqui, os princípios da reflexão estão muito claros. Toda vez que a bola bate contra uma borda, ela rebate no mesmo ângulo com que chegou. Isso acontece em qualquer lado da mesa.

O programa começa com a inicialização de algumas variáveis e com a escolha do modo gráfico adequado e das cores. O do TRS-Color também dimensiona três vetores, define dois UDG (Caracteres Definidos pelo Usuário) e os coloca nos vetores (um para a bola, um para a cruz e outro em branco). Nos programas do Apple e do MSX, usam-se duas formas predefinidas, uma para a bola e outra para a cruz. No MSX, isto é feito com Sprites, que são muito fáceis de definir e têm a vantagem de não apagar outros desenhos da tela que estejam em plano diferente. No Apple, utiliza-se uma tabela de formas. Não é tarefa simples montar uma delas sem o auxílio de um editor, mas o efeito final compensa. Para animar a figura, será preciso ter alguns cuidados (como apagar o desenho anterior).

As variáveis definidas são **BX**, **BY** (coordenadas **X** e **Y** para a bola); **CX**, **CY** (coordenadas para a cruz) e **NX** e **NY** (novas coordenadas para a cruz). **NX** e **NY** são as variáveis usadas para mover a cruz pela tela.

Depois desses procedimentos iniciais, um retângulo é desenhado na tela e a rotina **INKEY$** ou **GET**, que lhe permite mover o cursor, entra em ação. A rotina do TRS-Color usa **PEEK** nas posições de teclado, para possibilitar a auto-repetição das teclas.

Além de verificar a direção em que o cursor se move, a rotina checa também se alguma tecla está sendo pressionada. Caso haja pressão sobre a barra de espaços, o computador vai para a sub-rotina 500, que verifica a posição do cursor: se ele está sobre a bola, a força é imediatamente zerada e o computador retorna da sub-rotina. O TRS-Color não precisa especificar **P** (a variável para força) como 0, já que seu BASIC assume este valor para qualquer variável, a não ser que se determine algum outro.

Todo esse procedimento tem a finalidade de evitar a ocorrência de erros, pois, quando o cursor está sobre a bola, o computador não sabe em qual direção deve movê-la.

Se o cursor está numa posição válida, o fator força é definido como 0 (no Spectrum) ou 1 (nos outros micros). Isso ocorre no momento em que a barra horizontal é apagada.

TACADAS MAIS FORTES

A variável **P** é incrementada (tem seu valor aumentado) de 2, cada vez que o computador detecta uma pressão na barra de espaços. Quando atinge o valor 255, **P** volta a ser 0 — e a linha horizontal, que cresce com **P**, é novamente apagada. O computador permanece nesse círculo vicioso até que a barra de espaços se libere. No Apple, em vez da força zerar ao atingir o máximo, ela vai diminuindo aos poucos.

Assim que a condição da linha 540 deixa de ser verdadeira — ou seja, quando a barra de espaços é liberada — o computador continua a execução do programa que, neste ponto, varia um pouco de uma máquina para outra. O Spectrum e o TRS Color apagam a bola e traçam sua trajetória. O MSX e o Apple mostram a bola em movimento, deixando um rastro em sua trajetória. Mais adiante, explicaremos como funciona esta parte do programa.

Ao terminar os cálculos e o desenho do caminho da bola, o computador retorna para a linha 170. Em seguida, vai para a linha 300, no Spectrum, ou 220 nos outros micros.

O TRS-Color coloca a bola em sua nova posição e a cruz no mesmo lugar onde ela estava. As declarações **OR** que aparecem no fim de cada **PUT** evitam que a bola ou a cruz apaguem alguma coisa (pontos) da tela.

O computador espera que uma tecla seja pressionada para, então, apagar a tela e mostrar a velocidade inicial da bola e a distância percorrida. Em seguida, oferece mais uma tentativa. O Apple, novamente, apresenta uma diferença: a tela não é apagada, pois as informações são mostradas na janela de texto (últimas quatro linhas) da tela de alta resolução.

Como mostra a figura da página 653, para prever a direção que um objeto tomará após bater numa borda, basta estudar o ângulo com que atingiu a parede. Infelizmente, os computadores não dispõem de nenhum recurso para medir ângulos diretamente, precisando recorrer a uma equação.

CÁLCULO DAS COORDENADAS

Cada um dos programas define variáveis para as posições da bola e da cruz (há, de fato, dois conjuntos de coordenadas para a cruz, mas apenas um é usado aqui). O computador necessita de um outro conjunto — que contém a diferença entre as posições da cruz e da bola. De posse desses dados, pode calcular a direção da bola.

As variáveis que indicam a diferença entre as posições são **DX** e **DY**, calculadas na linha 560.

O programa utiliza muitas outras variáveis para os cálculos. O valor de **V**, variável usada para a velocidade inicial, é uma proporção da força que foi escolhida para impulsionar a bola (o valor exato desta proporção varia um pouco de um micro para outro, em razão de diferenças da tela).

RAÍZES QUADRADAS

A variável **SQ** iguala-se à raiz quadrada de (**DX*DX + DY*DY**). Esta é, na realidade, a distância entre a cruz e a bola, distância que é calculada por meio do teorema de Pitágoras, que estabelece que, para qualquer triângulo retângulo, o quadrado da hipotenusa (como é denominado o lado maior do triângulo) é igual à soma dos quadrados dos catetos (nome que recebem os outros dois lados).

A variável **T** corresponde ao intervalo de tempo usado nas equações. Se seu valor for maior, o programa funcionará mais rapidamente, porém, menos pontos serão traçados: como a razão entre o tempo e a distância fica menor, o número de pontos também diminui. Assim que se determina o intervalo de tempo, o computador começa a traçar os pontos da trajetória.

Em seguida, o programa utiliza mais duas variáveis para os cálculos. São os valores para a velocidade, nos eixos **X** e **Y** (ou horizontal e vertical). Em outras palavras, **VX** contém a velocidade com que a bola se desloca horizontalmente e **VY**, a velocidade de seu deslocamento vertical. As duas variáveis podem contar valores negativos: nesse caso, a bola se desloca da direita para a esquerda (para **VX**) ou de baixo para cima (para **VY**).

O valor dessas duas variáveis é calculado, como se pode verificar na linha 610, multiplicando-se a velocidade (**V**) pela variável **DX** (ou **DY**) e dividindo-se o resultado pela distância entre o centro da bola e o centro da cruz (variável **SQ**).

MICRO DICAS

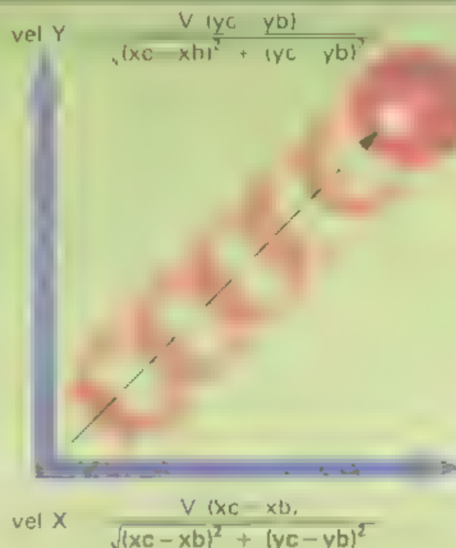
UTILIZAÇÃO DE MODELOS DINÂMICOS

Para quem não gosta muito de matemática ou física, pode parecer que os programas deste artigo não têm muitas aplicações práticas. Na verdade, estas são numerosíssimas.

Qualquer pessoa que tenha a aspiração de se tornar um programador competente — sobretudo nas áreas de jogos e programas educativos — precisa conhecer bem o mecanismo de simulação de modelos dinâmicos.

Veja, abaixo, alguns exemplos do que você pode fazer usando como base os programas deste artigo:

• um programa que simule um jogo de bilhar. Com o joystick ou o teclado, o jogador orienta o ângulo e a força de batida de um taco.
• um jogo de fliperama.
• um programa educativo que mostre como as moléculas se deslocam em um recipiente, colidindo continuamente umas com as outras. Quanto maior a densidade da matéria (por exemplo, densidade de gás), maior o número de colisões. Colisões geram calor; assim, a temperatura do gás pode ser estimada a partir de sua densidade ou pressão (esta é uma das leis fundamentais da física).
• um programa educativo para demonstrar o efeito de diferentes coeficientes de atrito, ou diferentes valores da força de gravidade sobre a movimentação livre de uma bola.



Como calcular a velocidade da bola

O programa recorre a várias equações para calcular a nova posição da bola e sua velocidade. As equações do diagrama separam a velocidade em suas componentes, com a projeção nos eixos **X** e **Y**. O computador resolve essas duas equações para cada posição da trajetória da bola, porém, sob uma forma um pouco diferente da que foi apresentada aqui. As constantes Xc − Xb e yc − Xb são substituídos no programa por **DX** e **DY**. Com elas, o computador calcula então a nova velocidade a cada ponto, levando em consideração a desaceleração.

Como mostra a fórmula, ele multiplica a diferença entre as coordenadas **X** e **Y** da bola e do cursor pela velocidade real, e divide o resultado pela linha de baixo da razão: a raiz quadrada da diferença entre as coordenadas **X** da bola e do cursor elevada ao quadrado somada à diferença das coordenadas **Y** da bola e do cursor elevada ao quadrado.

As duas equações seguintes calculam as novas coordenadas **X** e **Y**. Veja:

```
Xn = Xb + Vx*T − 0.01*T
Yn = Yn + Vy*T − 0.01*T
```

A nova posição da bola é calculada adicionando-se um número à antiga coordenada **X** e **Y**. Para calcular o número a ser adicionado, multiplica-se a velocidade no eixo **X** ou **Y** pelo tempo e, então, subtrai-se uma pequena fração de **T** ao quadrado, para levar em conta a desaceleração.

A velocidade é diminuída de uma pequena quantidade (0.1) e, antes de se reiniciar o *loop*, marca-se a nova posição da bola.

A NOVA POSIÇÃO DA BOLA

Neste ponto, o seu micro já dispõe de todos os valores necessários para calcular a nova posição da bola. A linha 620 faz esse cálculo para os novos valores de **BX** e **BY**.

O cálculo para cada uma das coordenadas é o seguinte:

```
BX = BX+VX*T-SNG(VX)*.1*T*T
```

Embora pareça complicado, não é. A fórmula adiciona um valor a **BX** (ou **BY**) para encontrar a nova coordenada. O número pode ser negativo, dependendo da direção em que a bola se desloque. Em vários pontos desta parte do programa para o Apple, você irá encontrar as variáveis **MX** e **MY** em lugar das variáveis **BX** e **BY**, porque o computador precisa das coordenadas antigas da bola para poder apagá-la, antes de desenhá-las em sua nova posição.

O primeiro passo é adicionar a **BX** a distância que a bola percorre. Esta distância é avaliada em dois estágios. Inicialmente, multiplica-se a velocidade pelo intervalo de tempo. Lembre-se de que a velocidade nada mais é do que a distância percorrida dividida pelo tempo gasto. Assim, para calcular a distância, basta multiplicar a velocidade pelo tempo.

Infelizmente, o programa precisa fazer um ajuste, pois as velocidades dadas pelas variáveis **VX** e **VY** não são iguais à velocidade real da bola em movimento — cada uma delas corresponde à velocidade de deslocamento em um eixo (horizontal ou vertical). Isso explica o estranho cálculo no fim da fórmula, envolvendo o sinal de **VX** e uma fração de **T** ao quadrado.

A função **SGN** fornece o resultado 1, -1 ou 0, conforme o argumento (número dentro dos parênteses) seja positivo, negativo ou zero. Ela é usada aqui para que se possa, empregando a mesma fórmula, subtrair o valor **.1*T*T** do termo anterior, no caso da velocidade no eixo ser negativa, ou somá-lo, no caso da velocidade ser positiva.

O resultado após cálculos desta linha corresponde à nova coordenada (para o eixo **X** ou **Y**) da bola. O cálculo é feito para os dois eixos.

COMO DESACELERAR

O programa diminui a velocidade de 0.1 e checa se ela é menor que 0.1. Em caso afirmativo, o micro vai para o fim da rotina, que é explicado abaixo. Em caso negativo, quatro linhas são usadas para verificar se alguma das bordas foi atingida no caminho para a nova posição (utilizam-se quatro **IF...THEN** para checar se as novas coordenadas estão caindo fora dos limites da mesa). Se a bola bate contra uma das bordas, o valor da variável **DX** (ou então **DY**, conforme a borda atingida tenha sido a superior ou a inferior) é igualado a menos **DX**.

Os valores de **BX** e **BY** também são alterados, de modo a colocar a bola na posição correta. Os números subtraídos das coordenadas antigas são o dobro dos limites da mesa. Assim, a maior coordenada da mesa no Spectrum é 127, o que significa que, toda vez que a bola encontra a borda superior, **BX** fica 254-**BX** (254 é o dobro de 127).

Depois das checagens, o computador calcula o próximo ponto. O fim da rotina, alcançado quando a velocidade se torna menor que 0.1, verifica se a bola não vai ser desenhada sobre a borda, antes de desenhá-la na tela.

Algumas das coordenadas usadas nesta verificação são diferentes daquelas usadas na anterior, porque o que vai ser desenhado é uma bola e não um ponto — e, é claro, a bola ocupa muito mais espaço. Assim, as coordenadas necessitam de algum ajuste.

SUGESTÕES

Você pode dar vários usos ao programa aqui apresentado. Uma idéia é transformá-lo em um jogo de *snooker*. Se esta for sua opção, bastará acrescentar algumas caçapas, para que as bolas caiam dentro, e uma rotina de contagem de pontos. Outra possibilidade é aproveitar esse programa para um jogo de *squash*, ou algo do tipo.

UM ASSEMBLER PARA O TRS-80

- MODIFICAÇÕES QUE DEVEM SER FEITAS NO PROGRAMA DO MSX
- CARACTERÍSTICAS DOS PROGRAMAS ASSEMBLY
- FUNCIONAMENTO DO ASSEMBLER

Com algumas modificações, o programa Assembler que apresentamos anteriormente para o MSX servirá também para os usuários do TRS-80. Veja aqui as adaptações necessárias.

Traduzir mnemônicos para código de máquina certamente é uma tarefa bastante cansativa. E, além de trabalhoso, colocar os códigos usando o monitor de seu TRS-80 pode dar origem a muitos erros — fatais, quando se trata de linguagem de máquina.

Neste artigo, você verá como utilizar o computador para fazer esse serviço. Com algumas modificações, o programa que publicamos anteriormente para o MSX servirá também para o TRS 80. Ele se destina a sistemas com 48K de memória, sendo que as funções de gravação e leitura dos programas em linguagem Assembly só funcionam em sistemas com disquetes. As outras funções do Assembler funcionarão normalmente em sistemas com fita cassete.

Como o programa foi escrito em BASIC, ele é um tanto lento, levando alguns minutos para montar um programa longo. Contudo, funciona muito bem.

Digite o Assembler para o **MSX** apresentado no artigo da página 401, com as seguintes modificações:



```
5000 CLS:PRINT@468,"Um instante
, por favor":CLEAR 500:DIM K$(1
10),K(110),M(110) H$="012345678
9ABCDEF":G$="0123456789abcdef"
5200 CLS PRINT@86,"ASSEMBLER":P
RINT:PRINTTAB(16)"(c)  carregar
 do disco":PRINT:PRINTTAB(16)"(
g)  gravar no disco" PRINT:PRIN
TTAB(16)"(e)  editar linha":PRI
NT:PRINTTAB(16)"(m)  montar":PR
INT:PRINTTAB(16)"(a)  apagar li
nha" PRINT:PRINTTAB(16)"(l)  li
star"
5210 A$=INKEY$:IFA$=""THEN5210
5220 JJ=INSTR("cgemal",A$)
5230 IFJJ=0THENPRINT"<"A$"> ???
" FORJ 1TO500:NEXT:GOTO5200
5240 CLS:ONJJGOSUB10420,10450,1
0490,5290,10710,10760
5250 PRINT:PRINT"Qualquer tecla
 para continuar"
5260 A$=INKEY$:IFA$=""THEN5260
5270 GOTO 5200
10420 INPUT"Nome do arquivo ",A
R$
10430 CLS:PRINT@465,"Carregando
 programa fonte"
10440 OPEN"I",1,AR$ INPUT#1,N:F
ORJ=1TON:INPUT#1,T$(J):NEXT:CLO
SE1 RETURN
10450 INPUT"Nome do arquivo ";A
R$
10460 CLS:PRINT@466,"Gravando p
rograma fonte"
10470 OPEN"O",1,AR$:PRINT#1,N:F
ORJ=1TON:PRINT#1,CHR$(34);T$(J)
;CHR$(34):NEXT:CLOSE1 RETURN
10490 PRINT"Qual o numero da li
nha (multiplo de 10) ";
10500 INPUTK:CLS
10510 K2=K/10:IFK2>NTHENK2=N+1:
N=N+1:T$(K2)="":PRINT@960,""
10520 IFK2<.1THENK2=.1
10530 IFK2=INT(K2)THEN10550
10540 K2=INT(K2)+1:FOR K3=NTOK2
-1STEP-1:T$(K3+1)=T$(K3):NEXT:N
=N+1:T$(K2)=""
10550 P1=16326:P0=P1
10560 PRINT@896,K;TAB(6)T$(K2)+
STRING$(20,32):P9=P0+LEN(T$(K2)
)
10570 IFP1<P0THENP1=P0
10580 IFP1>P9THENP1=P1-1
10590 FOR I=1 TO 1:POKE P1+I,3
2-(I=0)*99:NEXT I
10600 P7=0:A$=INKEY$:IFA$=""THE
N10600
10610 IFA$=CHR$(13)THEN10700
10615 IFA$=CHR$(27)THENRETURN
10620 IFA$=CHR$(9)THENP1=P1+1:G
OTO10580
10630 IFA$=CHR$(8)THENP1=P1-1:G
OTO10570
10640 IFA$=CHR$(10)THENA$="":GO
TO10670
10650 IFA$=CHR$(91)THENA$=" "+M
ID$(T$(K2),P1-P0+1,1):P7=-1:GOT
O10670
10660 IFA$<" "THEN10600
10670 IFP1-P0+1>LEN(T$(K2))THEN
T$(K2)=LEFT$(T$(K2),P1-P0)+A$:G
OTO10690
10680 T$(K2)=LEFT$(T$(K2),P1-P0
)+A$+RIGHT$(T$(K2),LEN(T$(K2))-
P1+P0-1)
10690 P1=P1-(LEN(A$)>0)+P7 GOTO
10560
10700 PRINT@64,"" K=K+10:GOTO 1
0510
10710 IFN=0THENCLS:PRINT"Nada a
 apagar":RETURN
10720 CLS:PRINT"Qual o numero d
a linha (multiplo de 10) ";
10730 INPUTK:K2=K/10
10740 IFK2>NORK2<1ORK2<>INT(K2)
THENPRINT"Esta linha nao existe
":RETURN
10750 K=K2 FORK3=K2TON:T$(K3)=T
$(K3+1):NEXT:N=N-1:PRINT@190,K*
10,"  ";T$(K):RETURN
10760 IFN=0THENPRINT"Nada a lis
tar":RETURN
10770 PRINT"Quais os numeros da
 primeira e da ultima linha (mu
ltiplos de 10)";
10780 INPUTK,K2:K=INT(K):K2=INT
(K2):K1=K/10 K2=K2/10
10790 IFK2>NTHENK2=N
10800 IFK1<1THENK1=1
10810 IFK2<K1ANDK2=NTHENRETURN
10820 IFK2<K1THENCLS PRINT"Nume
ros de linha invalidos":GOTO 10
770
10830 CLS:PRINT@192,:FORK3=K1TO
K2:PRINTK3*10" "T$(K3) NEXT
10840 RETURN
```

Algumas das linhas do programa — a 5450 e a 5750, por exemplo — são muito longas e podem dar problemas no momento de digitação. Para evitar que isso ocorra, digite as linhas exatamente como estão impressas, sem incluir espaços entre os comandos. Se mesmo assim o computador não aceitar uma linha devido ao seu tamanho, aperte **ENTER**, fazendo o começo da linha ir para a memória. Em seguida, edite a linha usando o comando **X** da edição.

Observe que nas linhas 5030, 5050, 5390 e 5500 encontram-se quatro caracteres em branco entre aspas; nas linhas 5660 e 5770, três.

Antes de montar o programa, proteja o topo da memória do computador para ali colocar os códigos. Faça-o logo que ligar a máquina — ou ativar o BASIC —, respondendo adequadamente à pergunta "Memória usada?".

COMO USAR O PROGRAMA

Ao executar o Assembler, um menu será exibido na tela. Para digitar um programa em Assembly tecle "e". No TRS 80, é necessário pressionar ao mesmo tempo as teclas **SHIFT** e @ (arroba) para usar as letras minúsculas. O computador perguntará, então, o número da linha. Cada instrução deve ficar em uma linha numerada, como acontece também no BASIC. Os números das linhas precisam sempre ser múltiplos de 10 e apenas um mnemônico Assembly com seus respectivos operandos deve ser introduzido em cada uma delas.

A primeira linha de um programa em Assembly especifica a origem ou endereço inicial do programa em código. Isto é feito pelo comando **org**, de modo que a primeira linha de seu programa ficará mais ou menos assim:

```
10 org -10000
```

Se os mnemônicos não forem escritos em letras minúsculas e os números não estiverem todos situados entre −32768 e 32767, haverá erro na execução do Assembler.

Os mnemônicos empregados seguem o padrão Z-80, com exceção de **ret**. Usado sozinho, este comando é aceito normalmente. Porém, nos casos de retorno condicional — **ret nz**, por exemplo — devemos usar **rts** em seu lugar.

Números hexadecimais precisam ser precedidos do sinal **$**; números binários, do sinal **%**. Números que não tenham sinal na frente serão interpretados como decimais.

Qualquer palavra que não corresponda a um mnemônico será interpretada como um rótulo. Evite palavras parecidas ou números.

Programas Assembly devem terminar com **end**, comando que indica o final da listagem.

Para editar uma linha — antes de apertar **ENTER** — utilizam-se as setas do teclado: "direita" e "esquerda" movem o cursor ao longo da linha; "para baixo" é usada para inserir e "para cima", para apagar caracteres. A edição das linhas é bem lenta. Muitas vezes, os caracteres digitados demoram a aparecer no vídeo.

Para sair do modo de edição, pressione simultaneamente as teclas **SHIFT** e "seta para cima". Elas devem substituir o **ENTER**, pois, se o utilizarmos depois que tivermos digitado a última linha, uma linha em branco será incorporada à listagem.

De volta ao menu, pressione "l", para obter uma listagem do programa em Assembly na tela.

Se houve algum erro durante a introdução do programa, volte ao menu e pressione "e"; em seguida, forneça o número da linha que deseja mudar.

Se quiser inserir uma linha entre duas já existentes, digite-a usando um número intermediário, no modo de edição. Quando listar o programa, verá que ele foi renumerado e que a nova linha ocupa o lugar certo. Porém, apenas uma linha pode ser introduzida dessa maneira de cada vez.

Para apagar uma linha, volte ao menu, tecle "a" e forneça o número da linha que quer eliminar.

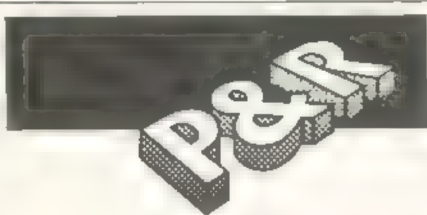
Quando estiver satisfeito com o programa Assembly (*programa-fonte*), volte ao menu e tecle "m" para que ele seja montado na memória. Ao mesmo tempo, o computador fornecerá uma listagem dos códigos hexadecimais equivalentes: o *programa-objeto*.

Uma vez montado o programa, o endereço final da rotina em código aparecerá na tela. Se for detectado algum erro durante a montagem, volte ao menu e edite a linha correspondente, repetindo, então, a montagem.

A opção "g" do menu grava o programa-fonte — os mnemônicos, portanto — em disco. A opção "c" faz a operação inversa.

Para gravar em fita cassete o programa que acaba de ser montado, você deve teclar **RESET**. Contudo, não pressione por muito tempo, nem mais de uma vez, sob pena de perder o programa-objeto e entrar no monitor de linguagem de máquina do TRS-80. Nos sistemas sem disquete não há como gravar o programa-fonte em Assembly.

Para executar o programa em código, use as instruções **DEF USR** e **A = USR(0)** (veja artigo da pág. 88).



O que é um Disassembler?

Como o próprio nome indica, o *Disassembler* executa uma função oposta ao *Assembler* — ou seja, enquanto o Assembler é um programa que traduz códigos mnemônicos (simbólicos) em código de máquina (números binários, diretamente executáveis pela UCP), o Disassembler reconstitui o programa em linguagem Assembly.

Como funciona um programa Disassembler? Da mesma maneira que no programa Assembler listado neste artigo, a base do processo de tradução é uma tabela. Só que o processo de desmontagem (disassembler) é mais complexo do que o de montagem, já que diversas instruções elementares do microprocessador Z-80 requerem dois ou mais bytes para serem totalmente armazenadas (ou seja, código de operação, mais operandos ou argumentos). Como o significado de um determinado byte na memória é diferente conforme o ponto da instrução em que ele se encontra, a ordem de leitura do programa em código de máquina é muito importante para o Disassembler. Uma única falha de interpretação de um byte invalidará toda a desmontagem até o final do programa.

O Disassembler não é um programa fácil de ser desenvolvido, principalmente em BASIC. Mas existem diversos programas Disassembler comercialmente disponíveis para o TRS-80.

MICRO DICAS

UTILIZAÇÃO DAS ROTINAS DA ROM NO SEU PROGRAMA EM ASSEMBLER

O Assembler é mais elementar do que o BASIC e outras linguagens de programação de alto nível. No entanto, é muito difícil utilizá-lo mesmo para a programação de tarefas simples pois ele exige que todas as operações da UCP, da memória, e de entrada/saída sejam especificadas da maneira mais minuciosa possível.

Mas existe um recurso capaz de aliviar esse cansativo trabalho: chamar as rotinas prontas que existem na memória ROM do computador.

Os micros da linha TRS-80 dispõem de um sistema operacional elementar e de um interpretador BASIC, ambos gravados na memória ROM. Esses programas contêm centenas de rotinas de operação do computador, que podem ser chamadas pelo comando **JSR** (desvio para sub-rotina), seguido do endereço onde elas começam.

Em geral, cada rotina necessita de um ou mais argumentos de entrada, e devolve um ou mais argumentos de saída. Os registros de intercâmbio de dados variam, mas comumente são os pares de registros internos do Z-80, como o HL e outros. Para utilizar as rotinas da ROM, você precisará, portanto, de um "mapa da mina". Existem vários livros que fornecem todos os endereços e argumentos.

SERRA PELADA: O TOQUE DE MIDAS

- ADICIONE AS ROTINAS QUE FALTAVAM
- LEVANTAMENTO GEOLÓGICO
- ESCAVAÇÕES
- GRÁFICOS ILUSTRATIVOS

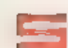
Figura da mitologia grega, o rei Midas recebeu de Dioniso (deus do vinho) o poder de transformar em ouro tudo o que tocava. Em nosso jogo, porém, as coisas não serão assim tão fáceis.



A primeira parte deste jogo apresentou opções como "Pesquisa e Desenvolvimento", "Levantamento Geológico", "Cavar mais 200 m", "Vender Ouro no Mercado" e "Passar a Vez". As subrotinas que se seguem completam o programa, particularizando cada opção.

A opção "Pesquisa e Desenvolvimento" é obtida por meio da tecla 1; "Levantamento Geológico" é a opção 2; "Cavar mais 200 m", a opção 3; "Vender Ouro no Mercado", a opção 4. A quinta opção cuida de "Passar a Vez", de forma que nenhuma rotina especial é necessária. As opções 1, 2 e 4 introduzem um elemento de acaso em seu resultado, tentando, assim, imitar a realidade do trabalho de mineração.

PESQUISA E DESENVOLVIMENTO



```
1000 BORDER 6: PAPER 6: INK 0:
CLS
1010 PRINT  PAPER 1; INK 6;AT 3
,4;" PESQUISA E DESENVOLVIMENTO
 ";AT 4,6;"(para reduzir os cus
tos)"
1020 PRINT AT 7,4;"Quanto prete
nde investir?($)": INPUT rd
1050 LET a(m,4)=a(m,4)-INT (rd*
.05)-1
1060 IF a(m,4)<0 THEN  LET a(m,
4)=0
1080 LET a(m,2)=a(m,2)-rd: LET
a(m,1)=a(m,1)-rd
1100 PRINT AT 13,3;"Os custos d
e mineracao foram";TAB 3;"reduz
idos em $";INT (rd*.05)+1;" por
 200 m"
1110 FOR z=1 TO 300: NEXT z
1120 RETURN
```



```
1000 GOSUB 1500
1010 LOCATE 1,0:PRINT "Pesquisa
 e Desenvolvimento":LOCATE 1,1:
PRINT " (para reduzir os custos
) "
1020 LOCATE 0,3:PRINT "Quanto p
retende":INPUT "investir ";RD
1030 IF RD<=0 THEN 1000
1050 A(M,3)=A(M,3)-INT(RD/20)-1
1060 IF A(M,3)<0 THEN A(M,3)=0
1080 A(M,1)=A(M,1)-RD:A(M,0)=A(
M,0)-RD
1100 PRINT:PRINT "Os custos de
mineração foram":PRINT "reduzid
os em $";INT(RD/20)+1;"por 200
m"
1110 FOR Z=1 TO 2000:NEXT
1120 RETURN
```



```
1000  HOME
1010  VTAB 1: HTAB 5: INVERSE :
 PRINT " PESQUISA E DESENVOLVIM
ENTO ": NORMAL : PRINT : PRINT
 TAB( 10)"P/REDUZIR OS CUSTOS"
1020  PRINT : PRINT : INPUT "QU
ANTO PRETENDE INVESTIR ";RD
1030  IF RD < 0 THEN 1000
1050 A(M,3) = A(M,3) -  INT (RD
 / 20) - 1
1060  IF A(M,3) < 0 THEN A(M,3)
 = 0
1080 A(M,1) = A(M,1) - RD:A(M,0
) = A(M,0) - RD
1100  PRINT : PRINT "OS CUSTOS
DE MINERACAO FORAM REDUZIDOS":
PRINT "EM $"; INT (RD / 20) + 1
;" POR 200 M"
1110  FOR Z = 1 TO 2000: NEXT
1120  RETURN
```



```
1000 CLS
1010 PRINT @2,"pesquisa e desen
volvimento":PRINT @34,"(PARA RE
DUZIR OS CUSTOS)"
1020 PRINT:INPUT"QUANTO PRETEND
E INVESTIR ";RD
1030 IF RD<0 THEN 1000
1050 A(M,3)=A(M,3)-INT(RD/20)-1
1060 IF A(M,3)<0 THEN A(M,3)=0
1080 A(M,1)=A(M,1)-RD:A(M,0)=A(
M,0)-RD
1100 PRINT @257,"OS CUSTOS DE M
INERACAO FORAM    REDUZIDOS EM
";INT(RD/20)+1;"POR 200 M"
1110 FOR Z=1 TO 2000:NEXT
1120 RETURN
```

No programa do Spectrum, a linha 1000, além de limpar a tela, muda suas cores. Nos demais computadores, as cores da tela permanecem as mesmas após a limpeza do vídeo.

A linha 1010 coloca um rótulo no alto da tela, antes que a linha 1020 pergunte ao jogador quanto dinheiro será investido em pesquisa e desenvolvimento (**RD**, ou **rd** no Spectrum, contém a quantia escolhida).

A linha 1050 diminui o custo de mineração de acordo com a quantia investida no processo de pesquisa. A linha 1060 assegura que o custo de produção não se torne negativo. A linha 1080 acerta o saldo total e a quantia do jogador, de acordo com o que foi gasto em "Pesquisa e Desenvolvimento".

A linha 1100 informa qual foi a redução de custos por 200 m de escavação. A linha 1110 contém um laço **FOR...NEXT** que provoca uma pequena demora antes que a sub-rotina termine.

LEVANTAMENTO GEOLÓGICO



```
2000 PAPER 4: BORDER 4: INK 0:
CLS
2030 LET r(m)=0: LET c(m,1)=INT
 (RND*90)+10: LET c(m,2)=INT ((
RND*5)+2)*200: LET c(m,3)=INT (
RND*200)+1: LET ll=INT (RND*3)-
1
2050 LET c(m,4)=c(m,2)+ll*200
2070 LET c(m,5)=0: LET kk=INT (
RND*100): IF kk<c(m,1) THEN  LE
T c(m,5)=1
2080 PRINT  PAPER 6: INK 0;AT 2
,5;" LEVANTAMENTO GEOLOGICO ":
PRINT AT 5,5;"Probabilidade de"
: PRINT AT 6,5;"encontrar ouro:
       ";c(m,1);"%": PRINT AT
8,5;"Provavel profundidade: ";c
(m,2);"m": PRINT AT 10,5;"Prova
vel quantidade: ";c(m,3);"kg"
2100 LET z=INT (RND*150000): LE
```

```
T a(m,2)=a(m,2)-z: LET a(m,1)=a
(m,1)-z
2110 PRINT  FLASH 1;AT 12,0;"De
seja  comecar a escavar (s/n)?"
2120 LET r$=INKEY$: IF r$="" TH
EN  GOTO 2120
2130 IF r$="s" THEN  LET a(m,6)
=0: LET r(m)=1: GOTO 3000
2500 RETURN
```



```
2000 GOSUB 1500
2030 R(M)=0:C(M,0)=INT(RND(1)*9
0)+10:C(M,1)=INT(RND(1)*5+2)*20
0:C(M,2)=INT(RND(1)*200+1):LL=I
NT(RND(1)*3)-1
2050 C(M,3)=C(M,1)+LL*200
2070 C(M,4)=0:KK=INT(RND(1)*100
):IF KK<C(M,0) THEN C(M,4)=1
2080 LOCATE 3,0:PRINT "Levantam
ento Geológico":PRINT:PRINT "Pr
obabilidade de":PRINT "encontra
r ouro:";C(M,0);"%"
2090 PRINT:PRINT "Provável":PRI
NT "profundidade:";C(M,1);"m":P
RINT:PRINT "Provável":PRINT "qu
antidade:";C(M,2);"kg"
2100 Z=INT(150000!*RND(1)):A(M,
1)=A(M,1)-Z:A(M,0)=A(M,0)-Z
2110 PRINT:PRINT "Deseja começa
r as":PRINT "escavações (S/N) ?
"
2120 R$=INKEY$:IF R$<>"S" AND R
$<>"N" THEN 2120
2130 IF R$="S" THEN A(M,5)=0:R(
M)=1:GOTO 3000
2500 RETURN
```



```
2000  HOME
2030 R(M) = 0:C(M,0) =  INT ( R
ND (1) * 90) + 10:C(M,1) = 200
*  INT ( RND (1) * 5 + 1):C(M,2
) =  INT ( RND (1) * 200 + 1):L
L =  INT ( RND (1) * 3) - 1
2050 C(M,3) = C(M,1) + LL * 200
2070 C(M,4) = 0:KK =  INT ( RND
 (1) * 100): IF KK < C(M,0) THE
N C(M,4) = 1
2080  VTAB 1: HTAB 7: INVERSE :
 PRINT " LEVANTAMENTO GEOLOGICO
 ": NORMAL : PRINT : PRINT : PR
INT "PROBABILIDADE DE": PRINT "
ENCONTRAR OURO: ";C(M,0);"%"
2090  PRINT : PRINT "PROVAVEL P
ROFUNDIDADE: ";C(M,1);" M": PRI
NT : PRINT "PROVAVEL QUANTIDADE
: ";C(M,2);" KG"
2100 Z =  INT ( RND (1) * 15000
0):A(M,1) = A(M,1) - Z:A(M,0) =
 A(M,0) - Z
2110  PRINT : PRINT "DESEJA COM
ECAR AS ESCAVACOES (S/N) ?"
2120  GET R$
2130  IF R$ = "S" THEN A(M,5) =
 0:R(M) = 1: GOTO 3000
2500  RETURN
```



```
2000 CLS
```

```
2030 R(M)=0:C(M,0)=RND(90)+9:C(
M,1)=(RND(5)+1)*200:C(M,2)=RND(
200):LL=RND(3)-2
2050 C(M,3)=C(M,1)+LL*200
2070 C(M,4)=0:KK=RND(100)-1:IF
KK<C(M,0) THEN C(M,4)=1
2080 PRINT @5,"LEVANTAMENTO GEO
LOGICO ":PRINT @129,"PROBABIL.D
E ENCONTRAR OURO";C(M,0);"%":PR
INT @193,"PROVAVEL PROFUNDIDADE
 ";C(M,1);"M":PRINT @257,"PROVA
VEL QUANTIDADE ";C(M,2);"KG"
2100 Z=RND(150000)-1:A(M,1)=A(M
,1)-Z:A(M,0)=A(M,0)-Z
2110 PRINT @353,"DESEJA COMECAR
 AS ESCAVACOES ? "
2120 R$=INKEY$:IF R$<>"S" AND R
$<>"N" THEN 2120
2130 IF R$="S" THEN A(M,5)=0:R(
M)=1:GOTO 3000
2500 RETURN
```

Todas as máquinas limpam a tela na linha 2000. O Spectrum também muda as cores da tela. A linha 2030 faz com que o valor de **R(M)** — r(m), no caso do Spectrum — seja zero, para indicar que as escavações ainda não começaram. Essa linha também calcula as possibilidades de encontrar ouro, além da quantidade de metal e da profundidade provável do veio **LL** (ou ll) é um numero aleatório entre −1 e 1 que será usado na próxima linha para determinar a profundidade real da mina (lembre-se de que o valor em **C(M,2)** é apenas a profundidade provável).

A linha 2050 faz com que **C(M,4)** assuma um valor igual a **C (M,2)** (mais ou menos 200 m, ou seja, 200 vezes **LL**). A seguir, a linha 2070 decide se a mina vai ou não ter ouro. Se não houver nenhum ouro, **C(M,5)** se tornará zero. **KK** é outro numero aleatório, entre 0 e 99. Ele é comparado com as chances de se encontrar ouro — se for menor do que estas, **C(M,5)** se tornará 1 para indicar que há ouro na mina.

A linha 2080 (2090 no Apple, no TK-2000 e no MSX) dá ao proprietário o resultado do levantamento geológico. Os números informam as chances de se encontrar ouro, onde e em que quantidade. Mas a existência ou não do metal na mina depende de uma série de fatores aleatórios. Dessa forma, o jogador deve ser muito criterioso ao fazer seus investimentos.

Agora, as más notícias: o relatório é pago e seu custo, imprevisível. Em todo caso, a conta fica no máximo em $150000 (valor de Z escolhido na linha 2100). O custo do relatório é então subtraído das posses do jogador, e essa dedução aparece no saldo total.

A seguir é oferecida a opção de iniciar a escavação — a linha 2110 pergunta: "DESEJA COMEÇAR AS ESCAVAÇÕES (S/N)?" Se a resposta for sim, o programa prosseguirá na subrotina de escavação da linha 3000.

A ESCAVAÇÃO

```
3000 BORDER 6: PAPER 6· INK 1:
CLS
3010 IF r(m)=0 THEN  PRINT  FLA
SH 1;AT 9,6;"A mina nao foi abe
rta!": FOR z=1 TO 10: SOUND .3,
-10: NEXT z: RETURN
3020 BORDER 5: INK 0: PAPER 4:
CLS
3022 PRINT  PAPER 5;TAB 14;CHR$
 147;CHR$ 148;CHR$ 149;TAB 14;C
HR$ 150;CHR$ 151;CHR$ 152;CHR$
153;TAB 13;CHR$ 154,CHR$ 155;CH
R$ 156;CHR$ 157;CHR$ 158;TAB 31
;CHR$ 32
3025 FOR z=1 TO 32· PRINT CHR$
144;: NEXT z
3060 PRINT AT 4,0;: FOR z=100 T
O 1400 STEP 100· PRINT TAB 4-LE
N STR$ z;z: NEXT z
3090 LET a(m,2)=a(m,2)-a(m,4):
LET a(m,1)=a(m,1)-a(m,4): LET a
(m,6)=a(m,6)+200: PAUSE 30
3100 PRINT AT 3,15;CHR$ 146: FO
R f=4 TO (a(m,6)/100)+3: PRINT
AT f,15;CHR$ 145: FOR w=1 TO 10
· SOUND .01,-20: NEXT w· NEXT f
3120 IF a(m,6)=c(m,4) AND c(m,5
)=1 THEN  GOTO 3500
3130 PRINT  FLASH 1; PAPER 5;AT
 6,6;" Nada de ouro ainda!": IF
 a(m,6)=c(m,2)+200 THEN  PRINT
 FLASH 1; PAPER 1; INK 6;AT 18,
0;" Nao ha ouro nesta mina.
     Va cavar em outro lugar.":
 FOR z=1 TO 10: SOUND .5,-20: N
EXT z: LET a(m,6)=0: LET r(m)=0
3140 PAUSE 150
3300 RETURN
3500 PRINT  PAPER 6; INK 2; FLA
SH 1;AT f,12;"O U R O": FOR z=-
20 TO 50: SOUND .017,z: NEXT z:
 PAUSE 75
3550 LET a(m,5)=a(m,5)+1: LET a
(m,3)=a(m,3)+c(m,3)· LET a(m,1)
=a(m,1)+(a(m,3)*er): LET a(m,6)
=0: LET r(m)=0: GOTO 3300
```

```
3000 CLS:GOSUB 1510
3010 IF R(M)=0 THEN PRINT:PRINT
:PRINT "A mina não foi aberta !
":PLAY "T25506V15AF":FOR Z=1 TO
 3000:RETURN
3020 LOCATE 14,3:PRINT CHR$(195
);CHR$(196);CHR$(197):LOCATE 14
,4:PRINT CHR$(198);CHR$(199);CH
R$(200);CHR$(201)
3022 LOCATE 13,5:PRINT CHR$(202
);CHR$(203);CHR$(204);CHR$(205)
;CHR$(206):LOCATE 31,5:PRINT CH
R$(32)
3025 LOCATE 0,6:FOR Z=0 TO 29:P
RINT CHR$(192);:NEXT
3060 LOCATE 0,7:FOR Z=100 TO 14
00 STEP 100:PRINT TAB(5-LEN(STR
$(Z)));Z·NEXT Z
3090 A(M,1)=A(M,1)-A(M,3):A(M,0
)=A(M,0)-A(M,4):A(M,5)=A(M,5)+2
00
3100 LOCATE 15,3:PRINT CHR$(194
):FOR F=4 TO (A(M,5)/100)+5:LOC
ATE 15,F:PRINT CHR$(193):PLAY"T
25501S12M1V15A64":NEXT F
3120 IF A(M,5)=C(M,3) AND C(M,4
)=1 THEN 3500
3130 LOCATE 4,0:PRINT " NADA DE
 OURO AINDA!":IF A(M,5)=C(M,1)+
200 THEN LOCATE 0,21:PRINT "Não
 há ouro nesta mina":PRINT "Vá
cavar em outro lugar":PLAY"T255
03CDEFG"·A(M,5)=0:R(M)=0
3140 FOR Z=1 TO 1500:NEXT
3300 RETURN
3500 LOCATE 10,0:PRINT " O U R
O"
3510 FOR Z=1 TO 10·PLAY"T25503C
A":NEXT
3550 A(M,4)=A(M,4)+1:A(M,2)=A(M
,2)+C(M,2)·A(M,0)=A(M,0)+(A(M,2
)*ER):A(M,5)=0·R(M)=0:GOTO 3140
```

```
3000  HOME
3010  IF R(M) = 0 THEN  VTAB 1·
 PRINT "A MINA NAO FOI ABERTA !
": CALL  - 198: FOR Z = 1 TO 10
00: NEXT : RETURN
3020  HGR : HCOLOR= 3: HPLOT 0,
50 TO 279,50
3030  HPLOT 125,45 TO 130,45 TO
 130,40 TO 125,40 TO 125,45: HP
LOT 145,45 TO 150,45 TO 150,40
TO 145,40 TO 145,45: HPLOT 135,
50 TO 135,40 TO 140,40 TO 140,5
0
3040  HPLOT 120,50 TO 120,40 TO
 130,30 TO 135,30 TO 135,25 TO
140,25 TO 140,30 TO 150,30 TO 1
50,5 TO 175,5 TO 175,35 TO 185,
45 TO 185,50 TO 210,50 TO 175,5
3050  HCOLOR= 6: FOR I = 58 TO
151 STEP 8: HPLOT 0,I TO 100,I:
 [illegible]
3090 A(M,1) = A(M,1) - A(M,3):A
(M,0) = A(M,0) - A(M,4):A(M,5)
= A(M,5) + 200
3100  HCOLOR= 5: FOR I = 5 TO (
A(M,5) * 8 / 100) + 50: HPLOT 1
55 + 4 *  RND (1),I TO 170 - 4
*  RND (1),I:X =  PEEK ( - 1633
6); NEXT
3120  IF A(M,5) = C(M,3) AND C(
M,4) = 1 THEN 3500
3130  VTAB 22: HTAB 1: PRINT "N
ADA DE OURO AINDA !": IF A(M,5)
 = C(M,1) + 200 THEN  PRINT "NA
O HA OURO NESTA MINA": PRINT "V
A CAVAR EM OUTRO LUGAR":A(M,5)
= 0:R(M) = 0
3140  FOR Z = 1 TO 1500: NEXT
3300  HGR : HOME : TEXT : RETUR
N
3500  HOME : VTAB 22: HTAB 15:
PRINT "O U R O !!!!": CALL  - 1
98
```

```
3550 A(M,4) = A(M,4) + 1:A(M,2)
 = A(M,2) + C(M,2):A(M,0) = A(M
,0) + A(M,2) * ER:A(M,5) = A(M,
5) = 0:R(M) = 0: GOTO 3140
```



```
3000 CLS
3010 IF R(M)=0 THEN PRINT @66,"
A MINA NAO FOI ABERTA!";:FOR Z=
1 TO 10:SOUND 120,1:NEXT:RETURN
3015 IF A(M,5)>0 THEN LINE(140,
40)-(157,191),PRESET,BF:GOTO 30
90
3020 PCLS:SCREEN 1,0:COLOR 3:LI
NE(0,0)-(255,31),PSET,BF
3022 PUT(131,8)-(168,31),H,PSET
3025 FOR Z=0 TO 31:PUT(Z*8,32)-
(Z*8+7,34),T,PSET:NEXT
3060 FOR Z=100 TO 1400 STEP 100
:Z$=STR$(Z)+"-":DRAW"C4S8BM"+ST
R$(49-8*LEN(Z$))+","+STR$(32+10
*Z/100):GOSUB 9000: NEXT
3090 SCREEN 1,0:A(M,1)=A(M,1)-A
(M,3):A(M,0)=A(M,0)-A(M,4):A(M,
5)=A(M,5)+200
3100 PUT (145,32)-(152,39),D,PS
ET:FOR F=4 TO (A(M,5)/100)+3:PU
T(145,F*10)-(152,F*10+9),B,PSET
:PLAY"T50O1BDBDEBDBDE":NEXT
3120 IF A(M,5)=C(M,3) AND C(M,4
)=1 THEN 3500
3125 FOR Z=1 TO 1000:NEXT
3130 PRINT @40,"NADA DE OURO AI
NDA!  ";:IF A(M,5)=C(M,1)+200 T
HEN PRINT @128,"NAO HA OURO NES
TA MINA. VA CAVAR EM OUTRO LUGA
R ";:PLAY"T50O3CDEFG":A(M,5)=0:
R(M)=0
3140 FOR Z=1 TO 2500:NEXT
3300 RETURN
3500 F=40+A(M,5)/10:COLOR 2:LIN
E(140,F)-(157,F+5),PSET,BF
3510 FOR Z=1 TO 10:PLAY"T50O2CA"
:PUT(140,F)-(157,F+5),H,NOT:NEX
T
3520 FOR Z=1 TO 2000:NEXT
3550 A(M,4)=A(M,4)+1:A(M,2)=A(M
,2)+C(M,2):A(M,0)=A(M,0)+(A(M,2
)*ER):A(M,5)=0:R(M)=0:GOTO 3300
```

Essa rotina é chamada de dois pontos diferentes do programa. A opção de cavar é oferecida após o levantamento geológico; mas ela também pode ser feita a partir do menu, aprofundando a mina em 200 m — opção 3.

Como de costume, a primeira linha limpa a tela (no Spectrum e no MSX, as cores mudam). A linha 3010 verifica se já foi feito um levantamento geológico. As linhas 3020 a 3090 cuidam dos gráficos que representam a mina. A linha 3100 ilustra o processo de escavação, ao mesmo tempo que produz alguns efeitos sonoros.

A linha 3120 verifica se as escavações atingiram o nível provável do filão e se existe ouro na mina (é possível que se tenha cavado mais de 1000 metros apenas para descobrir que não há nenhum metal). Se for achado ouro, o programa vai para a linha 3500, que dá as boas novas ao dono da mina, junto com alguns efeitos sonoros (menos no Apple e TK-2000). A linha 3550 ajusta as posses do jogador conforme a quantidade de ouro descoberta.

Se não houver ouro na mina, o programa continuará na linha 3130. Se a escavação ultrapassar o nível em que se esperava achar o metal (200 m), o jogador será informado de sua inexistência. Se o buraco ainda não chegou a essa profundidade, o jogador será informado: "NADA DE OURO AINDA!"

VENDA O OURO



```
4000 PAPER 6: INK 1: BORDER 6:
CLS
4020 PRINT  INVERSE 1;AT 2,4;"
CAIXA ECONOMICA FEDERAL ";AT 4,
5;"Agencia de Serra Pelada": PR
INT AT 8,1;" Cotacao do ouro: "
;er;" por kg";AT 14,3;"Quantos
quilos vai vender?": INPUT nte
4070 IF nte>a(m,3) THEN  PRINT
 FLASH 1;AT 18,4;"Voce nao tem
tanto ouro!"
4080 LET nte=INT nte
4090 IF nte>a(m,3) OR nte<0 THE
N  GOTO 4020
4095 PRINT AT 16,0;CHR$ 32;TAB
31;CHR$ 32
4100 LET a(m,3)=a(m,3)-nte: LET
 a(m,2)=a(m,2)+(nte*er): LET a(
m,1)=a(m,1)+(nte*er)
4130 PRINT  PAPER 5;AT 16,1;nte
;"kg vendidos por $";nte*er: PA
USE 170: RETURN
5000 RETURN
```



```
4000 GOSUB 1500
4010 LOCATE 2,0:PRINT " CAIXA E
CONOMICA FEDERAL":LOCATE 2,1:PR
INT" Agência de Serra Pelada"
4020 PRINT:PRINT "Cotação do ou
ro:":PRINT ER;"por kg de ouro":
PRINT:INPUT "Quantos kg vai ven
der";NT
4080 NT=INT(NT)
4090 IF NT>A(M,2) OR NT<0 THEN
4000
4100 A(M,2)=A(M,2)-NT:A(M,1)=A(
M,1)+NT*ER:A(M,0)=A(M,0)-NT*ER
4130 PRINT:PRINT NT;"kg vendido
s por";NT*ER:FOR Z=1 TO 2000:NE
XT
5000 RETURN
```



```
4000  HOME
4010  VTAB 1: HTAB 6: INVERSE :
 PRINT " CAIXA ECONOMICA FEDERA
L ": VTAB 2: HTAB 6: PRINT " AG
ENCIA DE SERRA PELADA ": NORMAL
4020  PRINT : PRINT "COTACAO DO
 OURO:": PRINT ER;" POR KG DE O
URO": PRINT : INPUT "QUANTOS KG
 VAI VENDER ? ";NT
4080 NT =  INT (NT)
4090  IF NT > A(M,2) OR NT < 0
THEN 4000
4100 A(M,2) = A(M,2) - NT:A(M,1
) = A(M,1) + NT * ER:A(M,0) = A
(M,0) + NT * ER
4130  PRINT : PRINT NT;" KG VEN
DIDOS POR ";NT * ER: FOR Z = 1
TO 1000: NEXT : RETURN
5000  RETURN
```



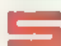
```
4000 CLS
4020 PRINT @4,"CAIXA ECONOMICA
FEDERAL":PRINT @136,"COTACAO DO
 OURO ":PRINT @197,"1 KG DE OUR
O=";ER:PRINT @228,"QUANTOS KG V
AI VENDER";:INPUT NT
4080 NT=INT(NT)
4090 IF NT>A(M,2) OR NT<0 THEN
4020
4100 A(M,2)=A(M,2)-NT:A(M,1)=A(
M,1)+(NT*ER):A(M,0)=A(M,0)+(NT*
ER)
4130 PRINT @448,NT;"KG VENDIDOS
 POR ";NT*ER:FOR Z=1 TO 1000:NE
XT:RETURN
5000 RETURN
```

A linha 4000 limpa a tela, como de costume. A linha 4020 coloca um rótulo no vídeo, imprime a cotação do ouro no mercado e pergunta quantos quilos serão vendidos. A linha 4070 verifica se o jogador tem realmente a quantidade de metal que pretende vender. A linha 4080 certifica-se de que essa quantidade é um número inteiro.

Se o jogador tentar vender mais do que tem, ou quiser comercializar uma quantidade negativa de ouro, a linha 4090 enviará o programa de volta ao início da sub-rotina. A linha 4100 ajusta as posses do jogador de acordo com o ouro vendido.

Finalmente, a sub-rotina informa ao jogador, por intermédio da linha 4130, a quantia que ele recebeu pela venda do ouro. A linha 5000 apresenta a opção de passar a vez.

RETOQUES FINAIS



```
   1 FOR n=USR "a" TO USR "o"+7
   : READ a: POKE n,a: NEXT n
7000 PAPER 5: INK 0: BORDER 5:
CLS
7010 PRINT AT 2,4;"JORNAL DE SE
RRA PELADA";AT 4,3;"Falencias
e Concordatas"
```

```
7020 PRINT AT 10,0:"   Faliu ho
je a empresa de mine-racao ";a$
(m);" Ltda,": PRINT "gracas a m
a administracao de seuproprieta
rio "
7025 PRINT  FLASH 1,AT 20,1:" Q
ualquer tecla para recomecar "
7030 PAUSE 0· RUN 5
8000 DATA 255.85,170.0,0.0,0,0,
62,28,56,126,28,62,120,28
8010 DATA 255.255,62,126.127,60
,124.126,0,0.0,0.1,1,1,1
8020 DATA 7,29.49.45,255.255,91
,126,128.96.48.80,152.140.252.1
38
8030 DATA 1.1,1,49,49,49.49.255
,122.187.62.95,153,255,153.126
8040 DATA 209,177,224,128,128.1
28.128.128.0,0,128,128,64,32,32
,16
8050 DATA 1.3.7.7.4.4.7.7.255 2
55,255,255,149.149.159.159
8060 DATA 24.126.153,255,126,15
3,126,219,128,192,224,240,248,1
68,248,255
8070 DATA 16,8,8,4,14,31,31,255
```



```
10 R=RND(-TIME)
70 SCREEN 1·COLOR 1,10,4·KEY OF
F
80 FOR I=0 TO 119
90 READ A·VPOKE BASE(7)+192*8+I
,A·NEXT
7000 GOSUB 1500
7010 LOCATE 2.0.PRINT" JORNAL D
E SERRA PELADA ":LOCATE 3,4:PRI
NT" FALENCIAS E CONCORDATAS"
7020 PRINT:PRINT "Faliu hoje a
emprêsa de"·PRINT "mineração ";
A$(M);" Ltda.":PRINT "graças à
má administração"·PRINT "de seu
 proprietário"
7030 LOCATE 0,19·PRINT "QUALQUE
R TECLA PARA RECOMEÇAR"
7040 IF INKEY$="" THEN 7040 ELS
E RUN
8000 DATA 255,85,170,0,0,0,0.0,
62,28,56,126,28,62,120.28
8010 DATA 255,255,62,126,127,60
,124,126,0,0,0,0,1,1,1,1
8020 DATA 7,29,49,45,255,255,91
,126,128,96,48,80,152,140,252,1
38
8030 DATA 1,1,1,49,49,49,49,255
,122,187,62,95,153,255,153,126
8040 DATA 209,177,224,128,128,1
28,128,128,0,0,128,128,64,32,32
,16
8050 DATA 1,3,7,7,4,4,7,7,255,2
55,255,255,149,149,159,159
8060 DATA 24,126,153,255,126,15
3,126,219,128,192,224,240,248,1
68,248,255
8070 DATA 16,8,8,4,14,31,31,255
```



```
7000  HOME
7010  VTAB 1: HTAB 7: INVERSE :
 PRINT " JORNAL DE SERRA PELADA
 ": NORMAL : VTAB 3: HTAB 7: PR
INT "FALENCIAS E CONCORDATAS"
7020  PRINT . PRINT "FALIU HOJE
 A EMPRESA DE MINERACAO": PRINT
 A$(M);" LTDA. GRACAS A MA ADMI
NISTRACAO": PRINT "DE SEU PROPR
IETARIO"
7030  PRINT   PRINT "QUALQUER T
ECLA PARA JOGAR NOVAMENTE"
7040  GET R$: RUN
```



```
7000 CLS
7010 PRINT @76,A$(M)·PRINT @168
,"ACABA DE FALIR! ":PRINT @449,
"QUALQUER TECLA PARA RECOMECAR"
7020 IF INKEY$="" THEN 7020 ELS
E RUN
9000 FOR K=1 TO LEN(Z$)
9010 B$=MID$(Z$,K,1)
9020 IF B$>="0" AND B$<="9" THE
N DRAW NU$(VAL(B$)):GOTO 9050
9030 IF B$="-"THEN DRAW "BF2R4"
9050 NEXT
9060 RETURN
```

As linhas 7000 a 7030 abrigam uma rotina para iniciar uma nova partida.

No Spectrum e no MSX, as linhas 8000 a 8070 contêm os dados para os blocos gráficos que desenham a mina e o processo de perfuração. No MSX, é usada a tela de textos de 32 colunas. Os padrões são guardados na tabela de padrões — **BASE(7)**. No TRS-Color, as linhas 9000 a 9060 desenham o buraco, no lado esquerdo da mina.

RECORRA AOS ARQUIVOS



ARQUIVOS SERIAIS

ARQUIVOS EM DISQUETE

A maior restrição para arquivos em fita cassete é que eles só podem ser lidos e gravados de forma serial; assim, eles funcionam na prática como um arquivo seqüencial. Felizmente, para as aplicações mais simples, isso não chega a ser um grande problema.

Os arquivos seriais são usados com muita freqüência, não somente em sistemas que funcionam apenas com fitas cassete, como também naqueles que empregam métodos versáteis de armazenamento em disquete.

Por outro lado, a maneira com que se lêem dados em disco permite que se utilize um outro tipo de arquivo: o *arquivo de acesso randômico* (também chamado de arquivo de acesso direto ou arquivo relativo). Os sistemas que trabalham com disco empregam ainda outros tipos de arquivo. Mas estes ou não são usados por micros domésticos ou são específicos de um determinado sistema operacional de disco.

Todo disquete conta com um certo número de blocos, onde são armazenadas as informações. Esses blocos se parecem com células de um favo de mel: ligadas entre si, mas claramente separadas. Os blocos são identificados por coordenadas: trilhas e setores. Isto permite que eles possam ser acessados diretamente e em qualquer ordem por um determinado programa.

Cada bloco pode ser lido, corrigido, regravado, sem que os outros sofram interferências. Desse modo, o processamento da informação se dá muito rapidamente e o tamanho do arquivo é limitado apenas pela capacidade de armazenamento do disco em uso.

ARQUIVOS NO FORMATO ASCII

Quando a informação arquivada é independente de programas e encontra-se armazenada separadamente, a quantidade de arquivos pode ser quase infinita. Com um programa editor de textos, por exemplo, tem-se arquivos separados para cartas padrão, artigos, avisos etc.

Uma das maiores vantagens de se trabalhar com um sistema de arquivos separados é que vários programas podem usar a mesma fonte de dados e não necessariamente do mesmo computador. Uma planilha de cálculo pode acessar arquivos de outros programas e um banco de dados padrão pode ser usado para armazenar informações destinadas a qualquer tipo de aplicação.

Isso pode ser feito por meio de arquivos seqüenciais escritos na liguagem padrão do código ASCII (para maiores esclarecimentos, veja o artigo *Trabalhe com o Código ASCII*, à página 361). A maioria dos computadores (mesmo aqueles que usam um código um pouco diferente do ASCII) permite que se opere com esses arquivos.

Neste caso, tudo que é gravado, incluindo comandos e variáveis, é transformado em seqüências de caracteres do código ASCII. Um programa acaba se transformando num arquivo de texto.

Uma vez gravada, a informação torna-se, pelo menos em teoria, acessível a outros computadores que possam ser ligados ao seu. Quando se usa um arquivo no formato ASCII, pode-se chamá-lo por intermédio de um editor de textos. Assim, todas as facilidades oferecidas pelo editor podem ser utilizadas para escrever o seu programa. E, quando estiver pronto, ele será gravado como um arquivo de texto.

Note que a "transportabilidade" de um arquivo no formato ASCII de uma para outras máquinas pode ser impedida se nele forem colocados caracteres de controle. Esses códigos são habitualmente usados pelo editor para controlar a impressora. E, como tais caracteres não são padronizados, provavelmente causarão problemas quando transferidos para outra máquina. Dessa forma um programa deve ser gerado sempre com a opção "não-documentação" ou "não formatação" (isto é, sem marcas de avanço de linha, parágrafo, negrito e outras, encontradas nos arquivos de texto a serem impressos como cartas.

ARQUIVOS DE DADOS

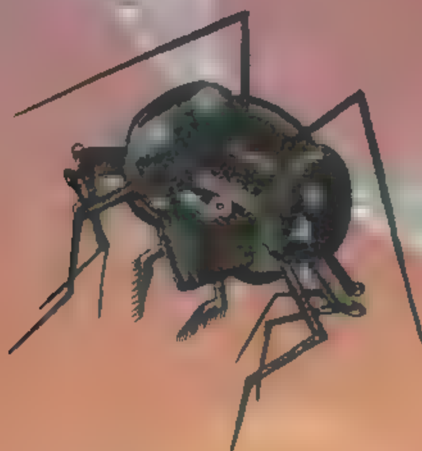
Qualquer programa que utilize um banco de dados — como um gerenciador de arquivos de dados, uma planilha de cálculo, ou os pacotes de contabilidade — funciona manipulando informações já fornecidas ao sistema. Ao se trabalhar com um programa desse tipo, seria extremamente cansativo ter que digitar toda a informação sempre que um aplicativo (como um programa de mala direta) fosse usado.

A solução para esse problema consiste em armazenar a informação separadamente do programa, como um arquivo de dados. Tal procedimento aumenta muito a capacidade de programa e diminui as exigências de memória do computador, visto que um sistema de armazenamento funciona como uma memória virtual, ficando na memória do computador apenas os dados em uso no momento.

O menor componente de um sistema de arquivamento — computadorizado ou não — é, de fato, a *entrada*. Várias entradas compõem um *registro*. Cada registro é dividido em *campos*. Cada campo tem um título ou rótulo, que permanece inalterado para todos os registros daquele arquivo. Esse rótulo serve para identificar o tipo de dado que o campo contém. Pode-se ver este tipo de estrutura em operação no programa apresentado no artigo *Organize as suas Coleções (1)*, à página 68.

Consideremos um arquivo simples de fichas. Ele é formado pelo conjunto completo das fichas, que são relaciona-







das por algum ponto comum (a lista dos sócios de um clube, por exemplo). Um registro é, neste caso, uma ficha usada para identificar apenas um membro do clube. Os campos são cada parte da informação contida na ficha — nome, endereço, telefone etc.

Em um sistema que funcione com papel, o arquivo pode ser transportado em um fichário, em pastas, caixas etc. Esses veículos, habitualmente chamados de arquivo, na realidade não passam de um modo físico de conter (e, às vezes, transportar) a informação.

Um sistema computadorizado funciona de modo bem diferente. A não ser que todos os dados estejam armazenados num só disco ou fita, não se estabelece aí a mesma relação "física" existente entre os pedaços de papel e seu invólucro. Em algumas aplicações, os dados ficam completamente separados, e podem ocupar discos ou fitas diferentes. Em outras, os dados são parte integrante do programa e não devem ser considerados um conjunto separado.

Arquivos separados podem, com frequência, ser usados por programas que não são responsáveis por sua criação. Assim, uma informação que foi trabalhada e armazenada por uma planilha de cálculo pode ser acessada por um editor de textos. Repare que a palavra é acessada e não transferida — ou seja, o programa lê e usa a informação sem removê-la. Num sistema manual, a transferência constitui a única forma de passar informação de um arquivo para outro sem copiá-la.

NOMES DE ARQUIVOS

O que não muda, no entanto, é a necessidade de se dar um nome a um arquivo. Seria difícil, se não impossível, acessar alguma informação num sistema computadorizado sem um nome com que o programa pudesse identificar rapidamente o conjunto de dados.

Para que um arquivo computadorizado seja adequadamente acessado por um programa, é preciso que seu nome seja escolhido com cuidado. Se não houver nomes, os dados armazenados em fita podem ser lidos na forma "carregar o próximo arquivo". Mas qualquer rotina de busca necessita de um nome pa-

ra identificar os dados requeridos.

Em fita cassete pode-se ter mais de um arquivo com a mesma denominação, ou mesmo arquivos sem nenhum nome. Mas esses procedimentos geram graves problemas. Se você instruir o computador para ler a última versão de um arquivo que tem o mesmo nome da cópia anterior, ele não poderá decidir que versão deve ler e ficará com a primeira que aparecer. A solução é dar a cada cópia um nome adequado e único.

Quando se usa um sistema de disco, o procedimento para a escolha de nomes é o mesmo. Nesse sistema, porém, é perfeitamente possível gravar um arquivo no lugar de outro (apagando este último), caso o mesmo nome tenha sido dado aos dois. Alguns sistemas operacionais de disco impedem que isso aconteça, mas, geralmente, deve-se proteger o arquivo de um apagamento acidental — "travando-o", como se diz no jargão da informática.

Ao escolher o nome, certifique-se de que não está ultrapassando o tamanho máximo permitido por cada micro. Em fita, o espaço para o nome de um arquivo não vai além de dez caracteres no Spectrum, oito no TRS-Color e seis no MSX. O Apple não usa nome para arquivos em fita. Em sistemas de disco, o tamanho varia bastante.

Convém ressaltar que o nome de qualquer arquivo computadorizado deve ser de fácil memorização. Além disso, ele precisa traduzir claramente seu conteúdo. Uma lista de todos os seus arquivos também ajudará bastante.

Os processos de transferência de dados através de um mecanismo de armazenamento são descritos pelas palavras gravar e ler (*write* e *read*, em inglês). Cada computador tem um conjunto diferente de comandos para essas duas operações, mas, de um modo geral, as rotinas são muito parecidas.

Normalmente, o programa principal é o primeiro a ser carregado na memória do computador — ele pode ser, por exemplo, um editor de textos ou um gerenciador de banco de dados.

Se o programa precisar de dados que não estão nele próprio, estes devem ser carregados de um arquivo de dados ou do teclado, manualmente. A partir daí, os dados serão corrigidos, alterados e trabalhados. Eles não poderão ser modificados enquanto permanecerem no disco ou na fita.

Em alguns arquivos, toda a informação tem que ser colocada na memória, mesmo que apenas uma pequena porção dela vá ser trabalhada. Este é o caso do Spectrum com um Microdrive. Em outros casos, pode-se ler apenas os dados que serão modificados. Por fim, a informação é gravada em um mecanismo externo de armazenamento (disquete ou fita cassete), ou enviada a outro mecanismo, como uma impressora.

Analisemos agora, mais detalhadamente, os estágios de gravação e leitura. Primeiro, deve ser dada ao computador uma instrução para abrir um canal de comunicação, de modo que ele saiba que a informação transitará por aquele ponto. Outras informações podem ser fornecidas a partir do comando de abertura do canal de comunicação, para especificar, por exemplo, que tipo de periférico está sendo acessado. Geralmente o periférico será um gravador cassete ou uma unidade de disquete.

Aberto o canal, são usados comandos para enviar ou gravar a informação no periférico escolhido. Inicialmente, os comandos são empregados para receber e não para enviar dados. Em qualquer dos casos, uma área de memória será utilizada para armazenamento temporário de informações (*buffer*).

Quando o acesso à informação termina, é preciso fechar o canal de comunicação. Assim, toda informação que estiver no meio do trajeto (ou eventualmente parada no buffer) será forçada a terminar o percurso. Para encerrar, o computador marca o fim do arquivo — *end of file* (eof), em inglês.

Até ser reaberto, o arquivo não transmitirá nem receberá nenhuma outra informação. Isso não impede a manipulação de arquivos por outros canais eventualmente abertos (sempre que o sistema seja capaz de trabalhar ao mesmo tempo com vários canais).

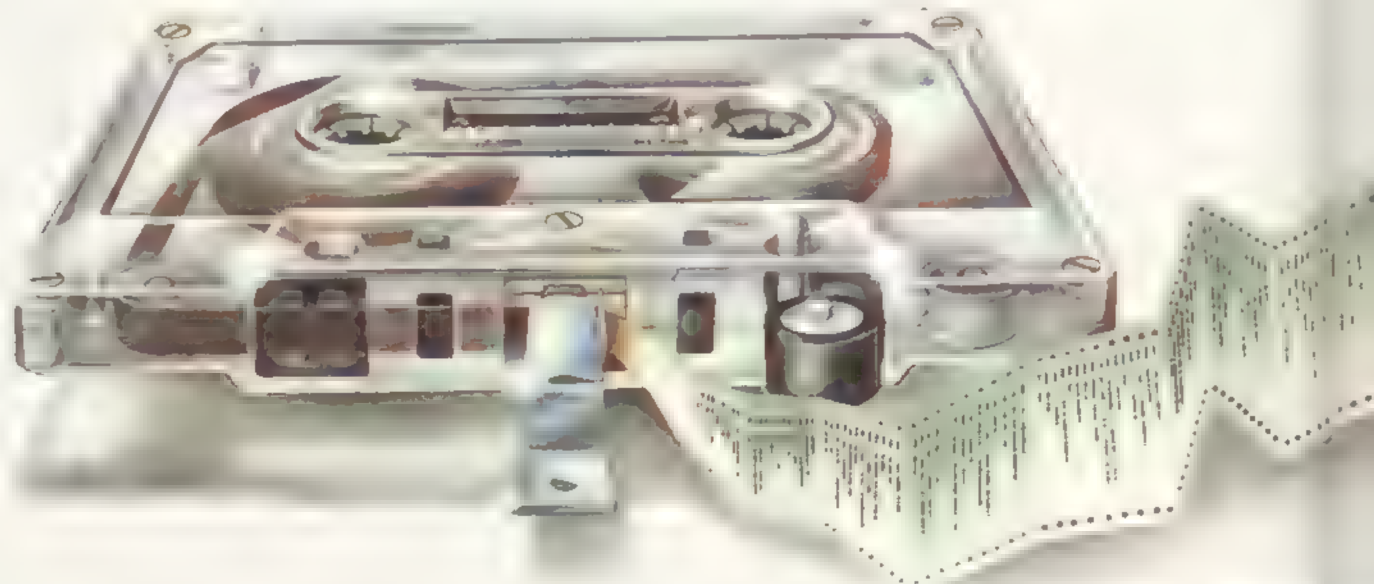
As instruções para executar essas tarefas variam conforme o micro. Vejamos cada uma delas. As instruções **SAVE** e **LOAD** (e variantes) que se aplicam a programas não serão abordadas.

Em cada um dos exemplos, a letra **N** se refere ao número do arquivo que é usado por muitos dos comandos; **X** (ou **X$**) é a informação transmitida.

S

Se você usa um gravador cassete para armazenar dados, está limitado a gravar programas, bytes e matrizes. Arquivos de dados podem ser manipulados com um Microdrive adequadamente instalado (esse dispositivo não é fabricado no Brasil). Os comandos de entrada e saída são os seguintes:

OPEN#



utilizado com a finalidade de preparar o sistema para a transferência de informação.

Em sua forma padrão, esse comando fica assim:

```
OPEN#N:"m",1:"NOME DE ARQUIVO"
```

onde a expressão **NOME DE ARQUIVO** pode ser uma variável previamente definida. Os dados, por sua vez, são transmitidos pelo canal de número **N**, sendo **m** o canal do Microdrive.

```
PRINT#
```

usado na forma **PRINT#N;X** (ou **X$**) com o objetivo de enviar a informação a ser gravada para o buffer de dados.

```
INPUT#
```

utilizado na forma **INPUT#N;X** (ou **X$**) para ler dados do buffer de dados. Estes são lidos até que surja um **CR** (*carriage return*).

```
CLOSE#
```

fecha um canal de comunicação previamente aberto. Seu formato é **CLOSE#N**.



Existem vários comandos para abrir linhas de comunicação, principalmente em um gravador cassete. Já as unidades de disco são acessadas por comandos que variam de acordo com o sistema operacional de disco em uso.

```
OPEN
```

abre um canal de comunicação para um periférico específico. Para gravar dados em fita, esse comando toma a seguinte forma:

```
OPEN "O",#-1,"NOME DE ARQUIVO"
```

onde apenas **NOME DE ARQUIVO** é uma variável definida pelo programa do usuário. O número -1 indica o gravador cassete. Se quisermos abrir um arquivo para leitura, devemos usar o comando na seguinte forma:

```
OPEN "I",#-1,"NOME DE ARQUIVO"
```

```
INPUT#
```

lê um cordão de um arquivo. A instrução **EOF(-1)** (eof significa *fim de arquivo*) pode ser usada para evitar erros (impede que se tente ler dados além da marcação de fim de arquivo).

```
PRINT#
```

envia informações para o periférico especificado pelo número N. O comando toma esta forma:

```
PRINT#N,X$
```

onde o caractere N é −1 para o gravador cassete e −2 para a impressora.
Já o comando

```
CLOSE#
```

fecha o canal de comunicação com o periférico. É usado na forma:

```
CLOSE#N
```

onde N é geralmente −1.



O MSX pode ser conectado tanto a um gravador cassete como a um controlador de discos para armazenamento de programas e dados. Abordaremos aqui os comandos referentes à manipulação de arquivos de dados em fita:

```
OPEN
```

utilizado para abrir um canal de comunicação com vários periféricos, que podem ser: gravador cassete, tela de texto, tela de gráfico, impressora, acionador de discos. Num arquivo em fita, o comando toma esta forma:

```
OPEN "CAS:NOME ARQUIVO"
 FOR INPUT AS #N
```

para se ler dados de um determinado arquivo usando o canal N (onde N pode ser 1, 2 etc.). Para gravação de dados, substitui-se **INPUT** por **OUTPUT**.

```
PRINT#
```

serve para enviar dados a um periférico. Toma a forma

```
PRINT#N,X (ou X$)
```

Quando se usa mais de uma variável na mesma instrução **PRINT #**, é necessário que sejam tomados alguns cuidados. Se forem variáveis numéricas, não há problema, pois o BASIC se encarregará de separá-las. No entanto, expressões alfanuméricas devem ser separadas explicitamente por vírgulas:

```
PRINT#1,A$;",";B$
```

Esse comando pode ser combinado com a instrução **USING** para determinar o formato do dado.

```
INPUT#
```

lê dados de um determinado periférico. Toma a forma

```
INPUT#N,X (ou X$)
```

No caso de variáveis numéricas, um espaço, uma vírgula, um retorno de carro (**CR**) ou um avanço de linha (**LF**) indicam o fim do número.

Para variáveis alfanuméricas, serão reconhecidos os marcadores acima (com exceção do espaço).

```
LINE INPUT#
```

é usado para fazer a leitura de variáveis alfanuméricas de um periférico. A diferença dessa instrução em relação ao comando anterior é que apenas o retorno de carro (**CR**) marca o fim do cordão que está sendo lido.

```
EOF(N)
```

indica se o final do arquivo foi encontrado. É habitualmente usado para evitar que se tente ler dados inexistentes no arquivo. O valor −1 retorna quando o marcador de fim de arquivo é encontrado. Esse comando é mais utilizado na forma

```
IF EOF(1) THEN instrução
```

onde **instrução** provoca geralmente o fechamento do arquivo.

```
CLOSE
```

serve para fechar o canal de comunicação de número N. Força o envio dos dados remanescentes no buffer de dados para o periférico ligado a esse canal. Toma a forma

```
CLOSE#N
```



O Apple e o TK-2000 são extremamente limitados em sua capacidade de manipular arquivos em fita. Como no Spectrum, apenas programas e matrizes podem ser gravados em fita. A seguir, descreveremos alguns comandos e aspectos mais importantes da manipulação de arquivos pelo sistema operacional DOS#3.3 do Apple.

Inicialmente, convém chamar a atenção para um fato essencial, que já custou muitas horas de trabalho a mais de um programador: todos os comandos do sistema operacional devem ser precedidos de um caractere especial — **CTRL-D**. Esse caractere deve estar sempre na primeira posição da linha. Desse modo, se antes de um comando houver um **PRINT** seguido de um ponto e vírgula, que deixe o cursor numa posição que não seja a primeira da linha, o comando não será reconhecido como do DOS. Igualmente importantes são as formas de colocar o **CTRL-D** na linha do comando. A mais segura consiste em definir uma variável que contenha esse caractere e colocá-la na linha de comando. Outra maneira é simplesmente acionar **CCTRL-D** (ou seja, pressionar a tecla **CTRL** simultaneamente com a tecla **D**) logo após ter aberto as aspas exigidas pelos comandos do DOS.

Nos exemplos a seguir, recorreremos ao primeiro método, usando a variável **D$** (freqüente entre os programadores do Apple). Ela é definida por meio do código ASCII de **CTRL-D:**

```
D$=CHR$(4)
```

Vejamos alguns comandos:

```
OPEN
```

abre um canal de comunicação entre a CPU e uma unidade de disco, assumindo a seguinte forma:

```
PRINT D$;"OPEN NOME ARQUIVO",
Dd, Ss, Vv
```

onde **NOME ARQUIVO** pode ser representado por uma variável alfanumérica e ter até trinta caracteres. A letra **D** é seguida do número do drive acessado (1 ou 2). A letra **S** é acompanhada do número do soquete (ou *slot*) no qual a placa que controla as unidades de disco está instalada (esse número é, em geral, 6). E **V** é seguido do número do volume do disquete, que é definido no momento de sua formatação. Esses três últimos parâmetros são opcionais e independentes: a presença de um não implica o uso dos demais.

```
READ
```

é empregado para habilitar um arquivo em disco para a leitura de dados e toma a seguinte forma:

```
PRINT D$,"READ NOME ARQUIVO"
```

A seguir, digita-se o comando **INPUT** para ler os dados desejados do arquivo em uso. O comando **PRINT D$**, por sua vez, encerra a leitura de informações do arquivo de dados (**D$** deve conter o caractere **CTRL-D**).

```
WRITE
```

envia e grava os dados em um arquivo em disco. Ele é usado na forma

```
PRINT D$;"WRITE NOME ARQUIVO"
```

Ele deve ser acompanhado de vários comandos **PRINT** para o envio dos dados. No caso de variáveis alfanuméricas, elas devem ser separadas por um **CR** (retorno de carro) para serem lidas individualmente mais tarde:

```
PRINT A$;CHR$(13);B$
CLOSE
```

fecha o canal de comunicação estabelecido. Ele é usado na forma

```
PRINT D$,"CLOSE NOME ARQUIVO"
,Dd, Ss, Vv
```

Como no comando **OPEN**, os três últimos parâmetros são opcionais. Neste caso, **NOME ARQUIVO** pode ser omitido, provocando o fechamento de todos os arquivos abertos no momento.

PROGRAMAÇÃO DE GRÁFICOS EM 3-D (4)

- OBJETOS MÚLTIPLOS
- VARIE A PERSPECTIVA
- COMO DESENHAR UM GLOBO
- CONTORNE O OBJETO
- DESENHE UM TORO



Agora que você já conhece os segredos da perspectiva, pode reunir vários problemas e rotinas e desenhar as mais diversas figuras flutuando no espaço interestelar.

Até aqui trabalhamos apenas com formas retangulares, quadrados e círculos concêntricos. Ao avançar um pouco mais em nosso estudo, entretanto, aprendemos a combinar (por meio de várias transformações) uma série de grades numa vista em perspectiva de um cubo. Este artigo ensina como pequenas mudanças no programa desenvolvido até este ponto pode nos tornar capazes de desenhar uma grande variedade de imagens tipo *wireframe*.

Começaremos com um desenho de figuras múltiplas. Se você já testou o primeiro programa do artigo *Programação de Gráficos em 3-D* (3), deve ter notado as alterações da imagem em perspectiva quando a distância do observador em relação ao objeto (D) é pequena, e o pouco efeito de perspectiva quando essa distância, ao contrário, é muito grande. Esse programa será desenvolvido agora de maneira a mostrar quatro cubos e não apenas um.

MULTIPLICIDADE DE OBJETOS

Os usuários que gravaram em fita ou disco os programas dos artigos precedentes desta série estão livres de boa parte do trabalho de digitação. Os que não fizeram isso terão que inserir não só o programa do artigo anterior (que se inicia na página 641) como também a rotina que desenha círculos, apresentada no primeiro artigo desta série (página 581).

O programa é constituído de blocos e rotinas, o que facilita a sua modificação para representar ao mesmo tempo quatro imagens em vez de uma.

Desse modo, se quiséssemos desenhar um conjunto de quatro cubos, por exemplo, precisaríamos especificar suas posições na tela e chamar quatro vezes a Rotina do Cubo. A posição estaria melhor especificada na rotina que transforma as coordenadas.

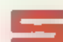
Passemos da teoria à prática. Para começar, modifique essa parte do programa como mostramos a seguir (mas não rode o programa por enquanto):



```
8500 X1 = T1 * X + T4 * Y + T7
+ XO
8510 Y1 = T2 * X + T5 * Y + T8
+ YO
8520 Z1 = T3 * X + T6 * Y + T9
+ ZO
```



```
8500 X1=T1*X+T4*Y+T7+XO
8510 Y1=T2*X+T5*Y+T8+YO
8520 Z1=T3*X+T6*Y+T9+ZO
```



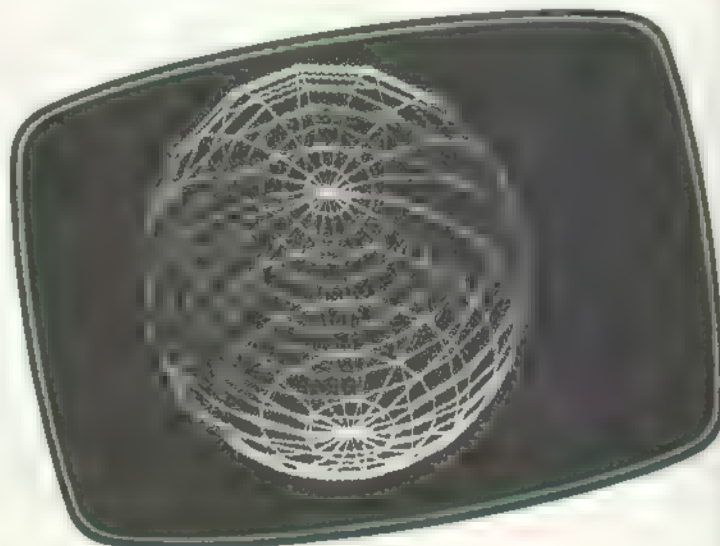
```
8500 LET X1=T1*X+T4*Y+T7+XO
8510 LET Y1=T2*X+T5*Y+T8+YO
8520 LET Z1=T3*X+T6*Y+T9+ZO
```

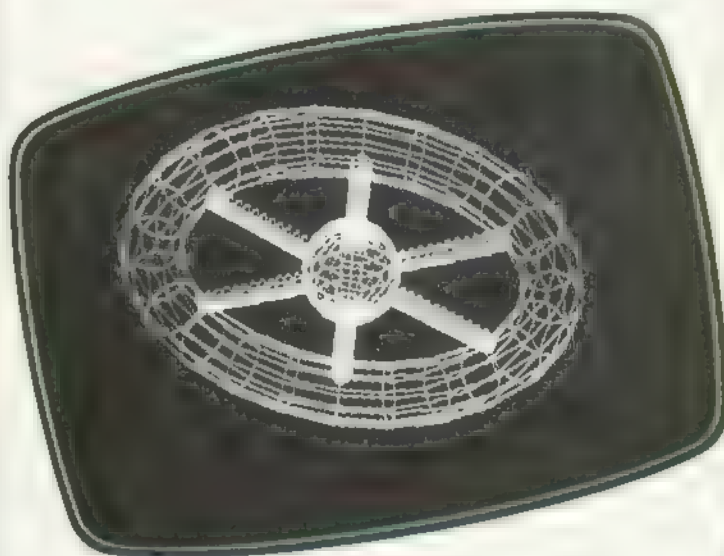
As variáveis **XO, YO** e **ZO** no final dessas três linhas especificam um equivalente, em cada uma das três coordenadas, para determinar a separação dos quatro cubos. São atribuídos valores a essas variáveis antes de os cubos serem desenhados.

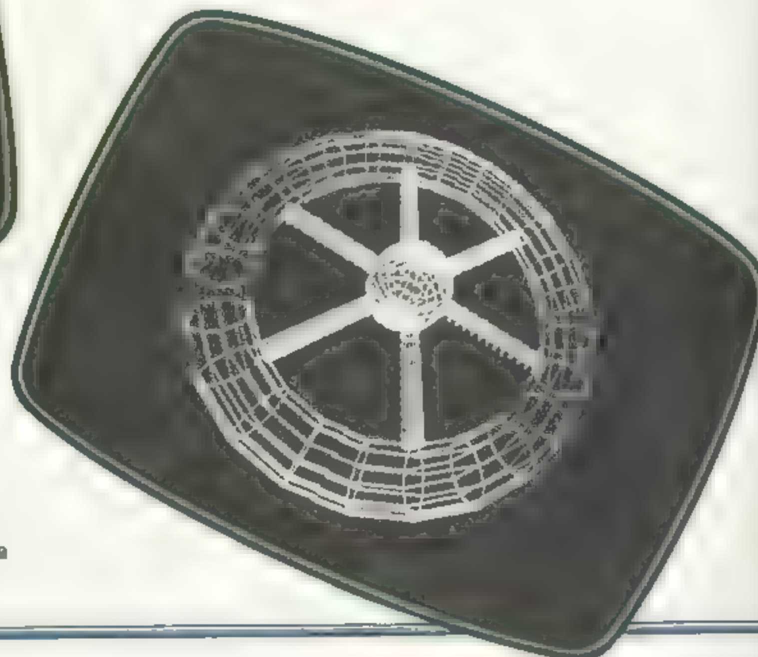
Agora digite essa outra seção de programa e rode-a para ver qual é o efeito do equivalente:



```
1520  GOSUB 1000
1530 XO = PP:YO =  - PP:ZO = 0
120 L = 20:PP = 20:N = 2
150  GOSUB 1500
1500 XO =  - PP:YO =  - PP:ZO =
 0
```







Globos, elipses, cubos, estações orbitais: as possibilidades criadas pelo programa do toro e pela rotina da grade são praticamente ilimitadas. Observe a deformação na perspectiva dos cubos, com pontos de fuga muito próximos.

```
1540  GOSUB 1000
1550 XO = PP:YO = PP:ZO = 0
1560  GOSUB 1000
1570 XO =  - PP:YO = PP:ZO = 0
1580  GOSUB 1000
1590  RETURN
```

```
120 L=10·PP=20:N=2
150 GOSUB 1500
1500 XO=-PP·YO=-PP·ZO=0
1520 GOSUB 1000
1530 XO=PP:YO=-PP:ZO=0
1540 GOSUB 1000
1550 XO=PP:YO=PP:ZO=0
1560 GOSUB 1000
1570 XO=-PP:YO=PP:ZO=0
1580 GOSUB 1000
1590 RETURN
```



A versão para o MSX é a mesma do TRS-Color, exceto pela linha:

```
120 L=20 PP=20·N=2
```



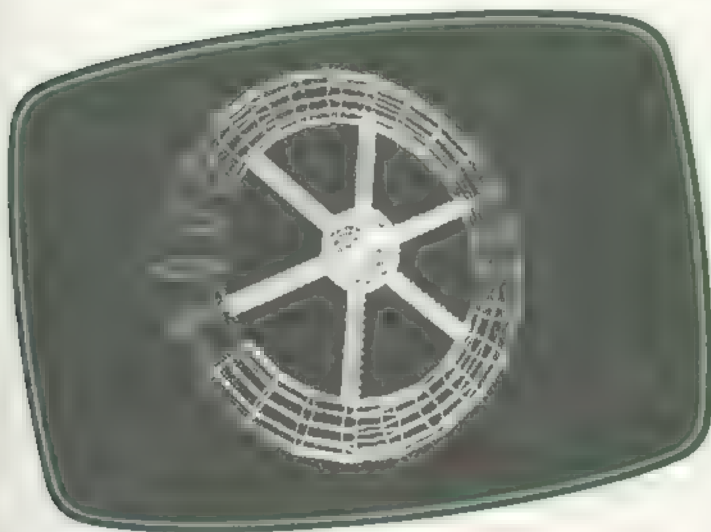
```
 120 LET L=20: LET PP=20: LET N
 =1
 150 GOSUB 1500
1500 LET XO=-PP: LET YO=-PP. LE
T ZO=0
1520 GOSUB 1000
1530 LET XO=PP: LET YO=-PP: LET
 ZO=0
1540 GOSUB 1000
1550 LET XO=PP: LET YO=PP: LET
ZO=0
1560 GOSUB 1000
1570 LET XO=-PP: LET YO=PP: LET
 ZO=0
1580 GOSUB 1000
1590 RETURN
```

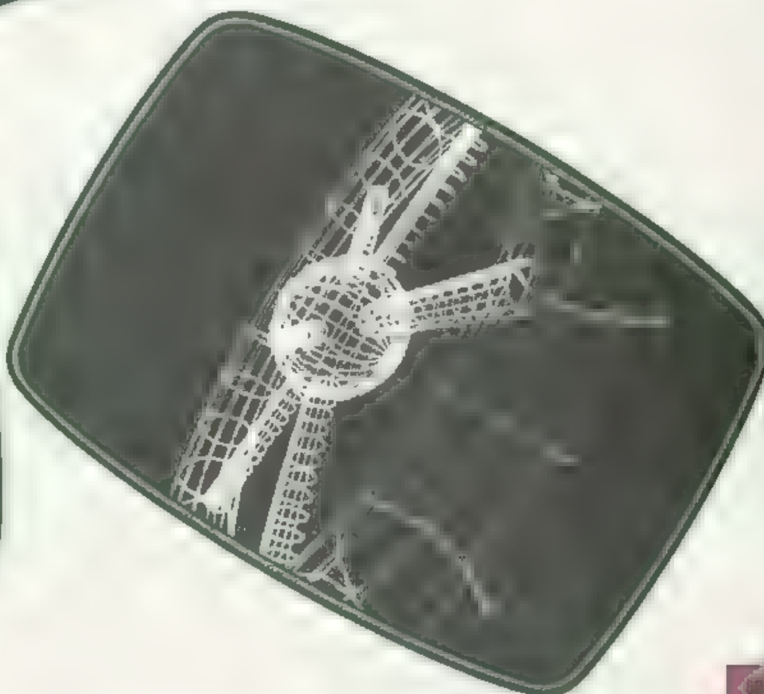
A linha 120 contém a variável **PP**, que transfere o valor para o equivalente na linha 1500.

A linha 150 chama uma rotina para desenhar quatro cubos. A sub-rotina especifica uma posição, como na linha 1530, por exemplo, e chama a rotina do cubo na linha seguinte, desenhando assim os quatro cubos.

Ao rodarmos o programa, teremos que fornecer alguns dados. O programa pedirá primeiro pelo valor de **D** (distância do plano de projeção). O valor 1000 será suficiente para começarmos. O programa então pedirá para que forneçamos a posição do observador (X, Y e Z). Experimente os valores que sugerimos a seguir e note como as imagens se distorcem quando **D** é pequeno (mas não quando ele é grande). Tente usar seus próprios valores também, incluindo milhares para **D** e números negativos ou zero para X, Y, e Z:







D	X	Y	Z
100	55	0	0
900	200	-250	200
20000	1000	3000	10000

A grande variedade de vistas possíveis com este programa nos permite fazer vários experimentos, mas não devemos nos limitar a modificar apenas os valores da posição do observador. A linha 120 contém variáveis responsáveis pelo comprimento dos lados do cubo **L**, pela separação destes (**PP**) e pelo número de quadrados na grade (**N**): três variáveis que podem ser alteradas antes de se rodar o programa.

DESENHE UM GLOBO

Embora curto, o programa que acabamos de inserir é capaz de desenhar imagens complexas, pois pode chamar qualquer uma das muitas rotinas contidas na listagem. Uma dessas rotinas poderia ser aquela que desenha círculos concêntricos (linhas 6000 a 6160), apresentada no artigo *Programação de Gráficos em 3-D (1)*, à página 581. Caso ela não esteja em seu programa, digite-a novamente e grave-a em disco ou fita para eventual uso futuro.

Em seguida, modificaremos o programa para que ele chame a rotina do círculo, em vez de chamar as rotinas da grade, do lado ou do cubo. Apague a linha 120 e acrescente estas linhas:



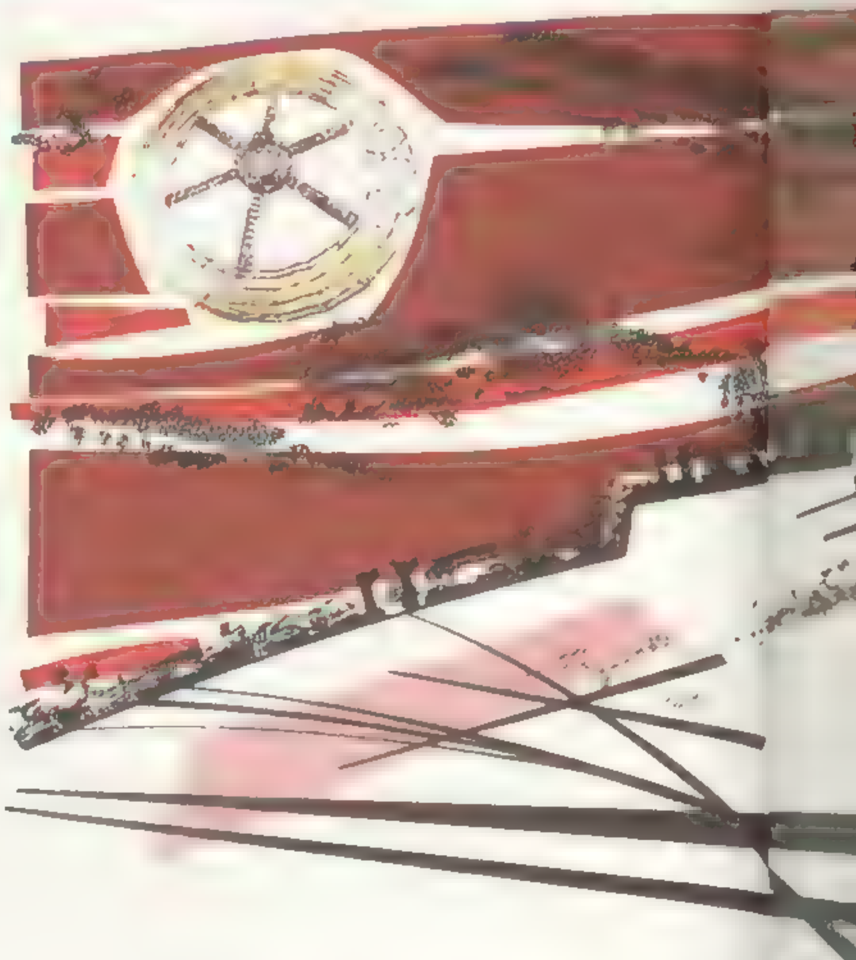
```
150 R = 20:N = 18: GOSUB 2000
2000 T7 = 0:T8 = 0:T9 = 0
2010 T4 = 0:T5 = 0:T6 = 1
2040 KA = 2 * PI / N
2050 KB = 0
2060  FOR KC = 1 TO  INT (N / 2
)
2070 T1 =  COS (KB):T2 =  SIN (
KB):T3 = 0
2080 XS = 0:YS = 0: GOSUB 6000
2090 KB = KB + KA
2100  NEXT KC
2110 T1 = 1:T2 = 0:T3 = 0
2120 T4 = 0:T5 = 1:T6 = 0
2130 KA = KA / 2
2140 KB = 0:RR = R
2150  FOR KC = 1 TO N
2160 T7 = 0:T8 = 0:T9 = RR *  C
OS (KB)
2170 XS = 0:YS = 0:R = RR *  SI
N (KB): GOSUB 6000
2180 KB = KB + KA
2190  NEXT KC
2200  RETURN
```



```
150 R=20:N=18:GOSUB 2000
2000 T7=0:T8=0:T9=0
2010 T4=0:T5=0:T6=1
2040 KA=2*PI/N
2050 KB=0
2060 FOR KC=1 TO INT(N/2)
2070 T1=COS(KB):T2=SIN(KB):T3=0
2080 XS=0:YS=0:GOSUB 6000
2090 KB=KB+KA
2100 NEXT KC
2110 T1=1:T2=0:T3=0
2120 T4=0:T5=1:T6=0
2130 KA=KA/2
2140 KB=0:RR=R
2150 FOR KC=1 TO N
2160 T7=0:T8=0:T9=RR*COS(KB)
2170 XS=0:YS=0:R=RR*SIN(KB):GOS
UB 6000
2180 KB=KB+KA
2190 NEXT KC
2200 RETURN
```



```
 100 LET XO=0: LET YO=0: LET ZO
=0
 150 LET R=20: LET N=18: GOSUB
2000
2000 LET T7=0: LET T8=0: LET T9
=0
2010 LET T4=0: LET T5=0: LET T6
=1
2040 LET KA=2*PI/N
2050 LET KB=0
2060 FOR K=1 TO INT (N/2)
2070 LET T1=COS KB: LET T2=SIN
KB: LET T3=0
2080 LET XS=0: LET YS=0: GOSUB
6000
2090 LET KB=KB+KA
2100 NEXT K
2110 LET T1=1: LET T2=0: LET T3
=0
2120 LET T4=0: LET T5=1: LET T6
=0
2130 LET KA=KA/2
2140 LET KB=0: LET RR=R
2150 FOR K=1 TO N
2160 LET T7=0: LET T8=0: LET T9
=RR*COS KB
2170 LET XS=0: LET YS=0: LET R=
RR*SIN KB: GOSUB 6000
2180 LET KB=KB+KA
2190 NEXT K
2200 RETURN
```



Ao rodar o programa, devemos fornecer valores para **D**, bem como para a posição do observador. Com esses valores, a linha 150 chama a rotina que acabamos de inserir e desenha um globo. Este é obtido por intermédio do de-





senho de uma série de elipses (ou círculos, dependendo da posição do observador). Tais elipses formam as linhas de longitude (que ligam os pólos do globo) e as de latitude, que correm paralelas à linha do equador.

O raio do globo é especificado em **R**, na linha 150, e o número de linhas de latitude e longitude, em **N**; este deve ser um valor par e maior do que 2. As linhas 2000 a 2050 ajustam as variáveis para posicionar o globo e para usá-las com contadores. O laço entre 2060 e 2100 desenha as linhas de longitude, e o que fica entre 2150 e 2190 traça as linhas de latitude.

O eixo do globo corta os pólos e sua direção é determinada pelas variáveis cujos valores definimos no começo do programa. Para verificar como os pólos do globo são posicionados em relação aos eixos da tela, estabeleça zero para duas das coordenadas da posição do observador (0, 0, 200, por exemplo). Depois, defina valores positivos ou negativos para as três coordenadas e veja como os pólos se movimentam em relação aos eixos.

DO GLOBO AO TORO

O globo é uma das formas mais simples; para desenhá-lo, recorremos a uma série de elipses, mas o mesmo programa contém todos os elementos necessários para o desenho de uma forma mais exótica — um toro (ou argola). Para realizá-lo, precisamos apenas de pequenas modificações na rotina do globo, como mostra a listagem a seguir:



```
150 R = 10:N = 18:RT = 20:N2 =
18: GOSUB 2000
2040 KA = 2 * PI / N2:NC = N2
2045  IF RT = 0 THEN NC =  INT
(NC / 2)
2060  FOR KC = 1 TO NC
2080 XS = RT:YS = 0: GOSUB 6000

2130 KA = 2 * PI / N
2135  IF RT = 0 THEN KA = KA /
2
2170 XS = 0:YS = 0:R = RT + RR
*  SIN (KB):N = N2: GOSUB 6000
```



```
150 R=10:N=18:RT=20:N2=18:GOSUB
 2000
2040 KA=2*PI/N2:NC=N2
2045 IF RT=0 THEN NC=INT(NC/2)
2060 FOR KC=1 TO NC
2080 XS=RT:YS=0:GOSUB 6000
2130 KA=2*PI/N
2135 IF RT=0 THEN KA=KA/2
2170 XS=0:YS=0:R=RT+RR*SIN(KB):
N=N2:GOSUB 6000
```



```
 150 LET R=10: LET N=18: LET RT
=20: LET N2=18: GOSUB 2000
2040 LET KA=2*PI/N2: LET NC=N2
2045 IF RT=0 THEN  LET NC=INT (
NC/2)
2060 FOR K=1 TO NC
2080 LET XS=RT: LET YS=0: GOSUB
 6000
2130 LET KA=2*PI/N
2135 IF RT=0 THEN  LET KA=KA/2
2170 LET XS=0: LET YS=0: LET R=
RT+RR*SIN KB: LET N=N2: GOSUB
 6000
```

Rode o programa e defina valores para **D** e para a posição do observador (1000, 200, 200, 200, por exemplo). Desta vez, **R** contém o raio interno do tubo, e o número de anéis é determinado por **N**. Esses anéis são equivalentes às linhas de longitude no globo. **RT** especifica a distância entre a origem e o tubo, e **N2** determina o número de linhas que formam o tubo, o que equivale às



Há alguma forma de se acelerar o desenho da estação espacial?

O modo mais grosseiro de se acelerar o programa consiste em reduzir o número de etapas nas rotinas principais — a rotina do círculo na estação espacial e a da grade nos cubos. Contudo, esse método reduz a suavidade das elipses e altera a estrutura do desenho. Uma alternativa melhor seria diminuir o número de acessos a essas rotinas.

Se quisermos, por exemplo, diminuir o número de segmentos que cortam o tubo de um toro, devemos reduzir os valores de **R**, **N**, **RT** e **N2** na rotina do toro que começa na linha 2000. É necessário também que as rotinas mais usadas estejam no começo do programa e que se usem variáveis em vez de números, sempre que possível. O uso indiscriminado de comandos **GOTO** também desacelera a execução de um programa.

Talvez a maneira mais fácil de acelerar esse programa seja omitir o desenho dos tubos que ligam o toro ao globo; faremos isso com um simples desvio (**GOTO**) no programa.

linhas de latitude no globo. O valor de **RT** deve ser maior que o de **R**, mas a rotina funcionará bem mesmo se isso não acontecer. Altere o valor dessas variáveis e observe o efeito obtido. Se **RT** for zero e **N** igual a **N2**, o toro se transformará numa esfera, como aquela da rotina do globo.

COMO COMBINAR IMAGENS

Não seria difícil substituir a rotina do cubo que desenha quatro imagens juntas na tela pela do globo ou do toro. Do mesmo modo, também seria simples chamar qualquer combinação das três rotinas e desenhar, por exemplo: cubos dentro de um globo ou de uma argola ou mesmo um sistema em que uma argola contenha um globo que, por sua vez, contenha um cubo. Poderíamos ainda usar declarações **GOTO** em certas linhas para suprimir os dois últimos lados, tornando assim menos confuso o desenho. Digite essas linhas e rode o programa para ver surgir uma imponente estação espacial.



```
150  GOSUB 3000
3000 R = 6:N = 12:RT = 40:N2 =
24: GOSUB 2000
3030  FOR F = 1 TO 6
3040 A = (F + .25) * PI / 3:L1
= 10:L2 = 34:N1 = 12:R = 2:N2 =
 8: GOSUB 3500
3050  NEXT
3060 R = 10:N = 12:RT = 0:N2 =
12: GOSUB 2000
3070  RETURN
3500 SA =  SIN (A)
3530 CA =  COS (A)
3540 T1 =  - SA:T2 = CA:T3 = 0
3550 T4 = 0:T5 = 0:T6 = 1
3560 EA = (L2 - L1) / N1
3570 EB = L1
3580  FOR EC = 0 TO N1
3590 T7 = EB * CA:T8 = EB * SA:
T9 = 0
3600 XS = 0:YS = 0:N = N2: GOSU
B 6000
3610 EB = EB + EA
3620  NEXT
3630 T1 = CA:T2 = SA:T3 = 0
3640  FOR EC = 1 TO N2
3650 EA = EC * PI / 8
3660 T7 = R * SA *  COS (EA):T8
 =  - R * CA *  COS (EA):T9 = R
 *  SIN (EA)
3670 XS = L1:YS = 0:XE = L2:YE
= 0: GOSUB 9500
3680  NEXT
3690  RETURN
```



```
150 GOSUB 3000
3000 R=6:N=12:RT=40:N2=24:GOSUB
 2000
3030 FOR F=1 TO 6
3040 A=(F+.25)*PI/3:L1=10:L2=34
:N1=12:R=2:N2=8:GOSUB 3500
3050 NEXT
3060 R=10:N=12:RT=0:N2=12:GOSUB
 2000
3070 RETURN
3500 SA=SIN(A)
3530 CA=COS(A)
3540 T1=-SA:T2=CA:T3=0
3550 T4=0:T5=0:T6=1
3560 EA=(L2-L1)/N1
3570 EB=L1
3580 FOR EC=0 TO N1
3590 T7=EB*CA:T8=EB*SA:T9=0
3600 XS=0:YS=0:N=N2:GOSUB 6000
3610 EB=EB+EA
3620 NEXT
3630 T1=CA:T2=SA:T3=0
3640 FOR EC=1 TO N2
3650 EA=EC*PI/8
3660 T7=R*SA*COS(EA):T8=-R*CA*C
OS(EA):T9=R*SIN(EA)
3670 XS=L1:YS=0:XE=L2:YE=0:GOSU
B 9500
3680 NEXT
3690 RETURN
```



```
150 GOSUB 3000
3000 R=6:N=12:RT=40·N2=24:GOSUB
 2000
3030 FOR F=1 TO 6
3040 A=(F+.25)*PI/3:L1=10:L2=34
:N1=12:R=2:N2=8:GOSUB 3500
3050 NEXT
3060 R=10:N=12:RT=0:N2=12:GOSUB
 2000
3070 RETURN
3500 SA=SIN(A)
3530 CA=COS(A)
3540 T1=-SA:T2=CA:T3=0
3550 T4=0:T5=0:T6=1
3560 EA=(L2-L1)/N1
3570 EB=L1
3580 FOR EC=0 TO N1
3590 T7=EB*CA:T8=EB*SA:T9=0
3600 XS=0:YS=0:N=N2:GOSUB 6000
3610 EB=EB+EA
3620 NEXT
3630 T1=CA:T2=SA:T3=0
3640 FOR EC=1 TO N2
3650 EA=EC*PI/8
3660 T7=R*SA*COS(EA):T8=-R*CA*C
OS(EA)·T9=R*SIN(EA)
3670 XS=L1:YS=0:XE=L2:YE=0:GOSU
B 9500
3680 NEXT
3690 RETURN
```



```
 150 GOSUB 3000
3000 LET R=6: LET N=24: LET RT=
40: LET N2=24: GOSUB 2000
3030 FOR F=1 TO 6
3040 LET A=(F+.25)*PI/3: LET L1
=10: LET L2=34: LET N1=12: LET
R=2: LET N2=8: GOSUB 3500
3050 NEXT F
3060 LET R=10: LET N=12: LET RT
=0: LET N2=12: GOSUB 2000
3070 RETURN
3500 LET SA=SIN A
3530 LET CA=COS A
3540 LET T1=-SA: LET T2=CA: LET
 T3=0
3550 LET T4=0: LET T5=0: LET T6
=1
3560 LET EA=(L2-L1)/N1
3570 LET EB=L1
3580 FOR E=0 TO N1
3590 LET T7=EB*CA: LET T8=EB*SA
: LET T9=0
3600 LET XS=0: LET YS=0: LET N=
N2: GOSUB 6000
3610 LET EB=EB+EA
3620 NEXT E
3630 LET T1=CA: LET T2=SA: LET
T3=0
3640 FOR E=1 TO N2
3650 LET EA=E*PI/8
3660 LET T7=R*SA*COS EA: LET T8
=-R*CA*COS EA: LET T9=R*SIN EA
3670 LET XS=L1: LET YS=0: LET X
E=L2: LET YE=0: GOSUB 9500
3680 NEXT E
3690 RETURN
```

A extensão de experimentos possíveis com o programa de toro é ilimitada. Tente alguns números pequenos para D (de 100 a 500, digamos), juntamente com diversos valores para X, Y e Z; você obterá, assim, diferentes perspectivas (especialmente porque os segmentos do toro que estiverem muito próximos serão omitidos). Tente também valores grandes para D e observe o que acontece com a distorção.

Agora, como simples desafio, poderíamos dar um toque final ao programa, adicionando cor ao fundo, por exemplo. Deixaremos por conta de sua criatividade incluir estrelas e planetas ao redor da estação.

MICRO DICAS

COMO MELHORAR A NITIDEZ

Quando a imagem dos quatro cubos é pequena - como acontece sempre que os valores para a posição do observador (X, Y e Z) são bem maiores que o da distância de observação D - torna-se difícil perceber o que se passa durante a execução do desenho. Pode-se melhorar a nitidez inserindo declarações **GOTO** em locais apropriados ao longo do programa (entre as linhas 1000 e 1170). A mais simples dessas declarações seria **GOTO 1220**, inserida na linha 1125 (linha nova) do programa dos quatro cubos.

ADIVINHAÇÃO DE PALAVRAS

- UM DESAFIO PARA DOIS
- MONTAGEM DA TELA
- AS REGRAS DO JOGO
- OS VALORES DAS LETRAS
- ESTRATÉGIA

Este jogo de adivinhação de palavras agrada a todas as idades, pode ter o nível de dificuldade que se queira e é muito versátil. Inspirado no "jogo da forca", destina-se a dois jogadores.

Alguns jogos para computador não são apenas recreativos — como os de aventura —, prestando-se muito bem para fins educativos. É o caso do que apresentamos aqui, inspirado no conhecido "jogo da forca".

Este, adaptado ou não para o computador, ajuda não só a ampliar o vocabulário do jogador, assim como seus conhecimentos gerais sobre vários temas (já que é usual a definição prévia de um assunto sobre o qual as frases ou palavras versarão).

O jogo de INPUT, [illegible] pessoas, também tem como objetivo a adivinhação de palavras ou frases [illegible] mais interessante e mais divertido [illegible] tradicional "jogo da forca" e tão educativo quanto ele. Pode ser jogado da mesma maneira, estabelecendo-se um assunto, o número de letras das palavras ou outro critério que se desejar.

O JOGO

Inicialmente, digite o nome dos dois jogadores. Em seguida, escolha o número de palavras da frase que cada jogador escreverá. Observe que, muitas vezes, frases mais longas são mais fáceis de se adivinhar, devido ao maior número de letras que aparecem.

Depois de ter escolhido o número de palavras, defina o número de jogadas que irão constituir a partida — isto é, quantas palavras cada jogador terá para adivinhar.

O primeiro jogador [illegible] pensar em uma frase e escrevê-la no computador. O adversário não precisa estar ausente enquanto você digita sua frase, pois as letras não aparecerão na tela. Mas, confiando em seu oponente, você poderá optar por ver as letras, o que lhe dará a certeza de que não cometeu nenhum erro de digitação.

Deixe apenas um espaço [illegible] cada das palavras da frase. No caso de um único vocábulo, nenhum espaço é permitido. O tamanho máximo de cada frase é de quarenta caracteres no micro Apple e no TK-2000, 64 no Spectrum [illegible] no TRS-Color e 78 no MSX.

Quando terminar a digitação, pressione a tecla <ENTER> para que a tela principal apareça. Em seu topo, [illegible] dará, então, o nome dos jogadores e o número de pontos de cada um — [illegible] para começar o jogo.

Abaixo do placar, há uma tabela indicando os valores das letras. As de uso freqüente têm valor mais alto [illegible] menos usuais têm um valor baixo. A frase a ser adivinhada aparece como uma seqüência de asteriscos.

No rodapé da tela encontram-se as instruções e, também, um espaço para os comandos e palpites.

ESTRATEGIA

O jogador terá três opções de jogo: comprar letras (ou espaço), adivinhar uma letra em determinada posição ou adivinhar a frase toda.

No início da jogada, uma boa opção é comprar um espaço — se suspeitar que a frase tem mais de uma palavra, é claro. Vogais são caras, mas aparecem com freqüência. As letras mais baratas aparecem muito menos e você corre o risco de não esclarecer nada com elas. Sempre é mais fácil chegar à frase correta quando se tem algumas consoantes — portanto, não se preocupe demais com as vogais.

Quando a frase começar a tomar forma, provavelmente você quererá adivinhar uma letra em determinada posição. Dada uma palavra como S*U, por exemplo, a letra O será um bom palpite. É nesse momento que se pode ganhar pontos. Colocando uma letra na posição correta, você ganha o seu valor em pontos. Se errar, perde apenas metade do valor. Pressione XX para selecionar esta opção e dar o seu palpite.

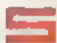
Você pode, também, ter uma súbita inspiração e tentar acertar a frase toda. Nesse caso, deve pressionar ZZ e escrever a frase. Se ela estiver correta, o valor de todas as letras que ainda não tinham sido descobertas será acrescentado ao seu placar. Se você errar, perderá 50 pontos.

Digite a primeira parte do programa. [illegible]



tar todos os dados para iniciar o jogo. Mas com ela você não irá muito longe. A parte final — que contém as rotinas que lêem os palpites, fazem as correções e contam os pontos — fica para o próximo artigo.

Não se esqueça de gravar o programa.



```
10 LET R$="PALAVRA": LET W=14
: LET d=0: LET f=1: LET o$=""
: LET g=0: LET k=0: LET q$=""
: LET ta=200: LET tb=200: LET
tc=0: LET b=0: POKE 23609,50·
POKE 23658,8: LET i$="": LET
j$=""· LET z$="": LET c$=""
20 FOR n=0 TO 7: READ y: POKE
USR "a"+n,y· NEXT n
30 DATA 255,129,129,129,129,
129,129,255
40 INPUT "NOME DO PRIMEIRO JO
GADOR ?        (ATE 7 LETRAS)",
 LINE a$
50 INPUT "NOME DO SEGUNDO JOG
ADOR ?         (ATE 7 LETRAS)"
, LINE b$
60 IF LEN a$>7 OR LEN b$>7
THEN  GOTO 40
70 CLS : INPUT "QUANTAS PALAV
RAS POR FRASE?(1-9)", LINE c$
 80 IF LEN c$<>1 THEN  GOTO 70
 90 IF CODE c$<49 OR CODE c$>
 57 THEN  GOTO 70
100 LET c=VAL c$
110 INPUT "NUMERO DE JOGADAS ?
(1 a 9)", LINE t$
120 IF LEN t$<>1 THEN  GOTO
110
130 IF CODE t$<49 OR CODE t$>
57 THEN  GOTO 110
140 LET t=VAL t$
150 IF c>1 THEN  LET j$="S":
LET i$="COM UM ESPACO ENTRE CA
DA": LET r$="FRASE"
160 PRINT a$;",E SUA VEZ DE JO
GAR."'''"INTRODUZA SUA FRASE DE
";c;" PALA - VRA";j$;".AS LETR
AS QUE VOCE INTRODU-ZIR SERAO
INVISIVEIS,MAS SE VOCEQUISER V
ER ENTAO PRESSIONE '0'. PARA C
ONTINUAR, PRESSIONE 1."
170 LET k$=INKEY$: IF k$=""
THEN  GOTO 170
190 IF k$="0" THEN  POKE 23624
,56: INPUT  LINE s$: CLS :
GOTO 220
200 IF k$="1" THEN  POKE 23624
,63· INPUT  LINE s$· CLS :
POKE 23624,56: GOTO 220
210 GOTO 170
220 LET l=LEN s$
230 IF l=0 THEN  PRINT "ENTRAD
A ILEGAL. TENTE DE NOVO":
PAUSE 100: CLS : GOTO 160
240 IF l>64 THEN  PRINT "ENTRA
DA MUITO LONGA. TENTE DE   NOV
O": PAUSE 100: CLS : GOTO 160
250 FOR n=1 TO l: IF s$(n)=
CHR$ 32 THEN  LET d=d+1: GOTO
270
260 IF CODE s$(n)<65 OR CODE
s$(n)>90 THEN  PRINT "LETRA IL
EGAL. TENTE DE NOVO": PAUSE
100: CLS : LET d=0: GOTO 160
270 IF c=1 AND d=1 THEN  PRINT
"NAO HA ESPACOS DENTRO DE UMA
   UNICA PALAVRA. TENTE DE NOV
O.": PAUSE 100: CLS : LET d=0:
 GOTO 160
280 NEXT n
290 IF d<>c-1 THEN  PRINT "VOC
E DEVE INTRODUZIR ";c;" PALAVR
AS ";i$;". TENTE OUTRA VEZ":
PAUSE 100: CLS : LET d=0: GOTO
 160
300 LET z$=""
310 FOR n=1 TO l· LET z$=z$+"*
"· NEXT n
320 PRINT  INK 1,AT 0,0,"SCORE
/";a$: PRINT  INK 1,AT 0,16,"S
CORE/";b$: PRINT  PAPER 2, INK
6;AT 1,6;ta;TAB 22;tb,TAB 31,"
"
330 PRINT AT 3,7;"VALOR DOS CA
RACTERES"
340 FOR n=0 TO 26· READ o$·
LET q$=q$+o$· NEXT n· PRINT q$
· RESTORE 900
350 PRINT  INK 1;AT 12,0;"A ".
r$;" QUE ";b$;" TEM QUE  ADIVI
NHAR CONTEM ";L;" CARACTERES":
PRINT  PAPER 2; INK 6;z$
360 INPUT "VOCE QUER COMPRAR U
M CARACTER   PELO PRECO EXIBID
O NA TABELA?   FACA A ESCOLHA
DO CARACTER.     SENAO, TECLE
XX PARA ADIVINHAR  UM CARACTER
 OU ZZ PARA ADIVINHARA FRASE T
ODA.", LINE d$
1000 DATA "A-20    ","B-10    "
,"C-10    ","D-12    ","E-20
","F-08    ","G-12    ","H-08
"
1010 DATA "I-20    ","J-04    "
,"K-06    ","L-10    ","M-10
","N-10    ","O-20    ","P-10
 ","Q-02    ","R-12    ","S-1
2    "
1020 DATA "T-12    ","U-20    "
,"V-08    ","W-08    ","X-04
","Y-08    ","Z-02    ","<GRAP
HICS A>-20    "
```

TT

```
5 CLEAR 1000
10 R$="PALAVRA":W=14:F=1:TA=200
:TB=200
15 P1=PEEK(359):P2=PEEK(360):P3
=PEEK(361)
40 CLS:LINE INPUT"NOME DO PRIME
IRO JOGADOR         (MAX 7 LETRA
S) ?";A$
50 PRINT·LINE INPUT"NOME DO SEG
UNDO JOGADOR         (MAX 7 LET
RAS) ?";B$
60 IF LEN(A$)>7 OR LEN (B$)>7 T
```

MICRO DICAS

MANIPULAÇÃO DE CORDÕES

Os comandos BASIC capazes de agir sobre variáveis alfanuméricas constituem a chave do sucesso do programador interessado em jogos e aplicativos que manipulem palavras.

Comandos como **RIGHT$**, **LEFT$** e **MID$** permitem a "leitura" do conteúdo de uma cadeia de caracteres. E temos ainda funções como **LEN**, **CHR$**, **STR$** e **STRING$**, que tornam possível a criação de novos cordões ou a transformação dos antigos.

Observe que os comandos seguidos por um cifrão —"$"— geram novos cordões, enquanto os comandos sem este sinal resultam num valor numérico que corresponde ao operando.

```
HEN 40
70 CLS:LINE INPUT"ESCOLHA O NIV
EL DE DIFICULDADE  (NUMERO DE P
ALAVRAS/FRASE 1-9) ?";C$
80 IF LEN(C$)<>1 THEN 70
90 IF C$<"1" OR C$>"9" THEN 70
100 C=VAL(C$)
110 PRINT:LINE INPUT "NUMERO DE
 JOGADAS (1-9) ? ";T$
120 IF LEN(T$)<>1 THEN  110
130 IF T$<"1" OR T$>"9" THEN 11
0
140 T=VAL(T$)
150 IF C>1 THEN J$="S":I$="COM
UM ESPACO ENTRE CADA":R$="FRASE
"
155 CLS
160 PRINT A$;". E SUA VEZ DE JO
GAR":PRINT:PRINT"INTRODUZA SUA
FRASE DE ";C;" PALA-VRA";J$
165 PRINT:PRINT"PRESSIONE'0'PAR
A VER AS LETRAS  OU '1' PARA CO
NTINUAR ...":PRINT
170 K$=INKEY$:IF K$="" THEN 170
190 IF K$="1" THEN PRINT "? ";:
POKE 359,&H86:POKE 360,32:POKE
361,57:LINE INPUT S$:POKE 359,P
1:POKE 360,P2:POKE 361,P3:GOTO
220
200 IF K$="0" THEN LINE INPUT "
? ";S$:GOTO 220
210 GOTO 170
220 L=LEN(S$):PRINT
230 IF L=0 THEN PRINT "ENTRADA
ILEGAL - TENTE DE NOVO":GOSUB 9
50.CLS:GOTO 160
240 IF L>64 THEN PRINT"ENTRADA
MUITO LONGA-FACA DE NOVO":GOSUB
 950:CLS:GOTO 160
250 FOR N=1 TO L:IF MID$(S$,N,1
)=CHR$(32) THEN D=D+1:GOTO 270
260 IF MID$(S$.N.1)<"A" OR MID$
(S$,N,1)>"Z" THEN PRINT "CARACT
ER ILEGAL - TENTE DE NOVO":GOSU
B 950:CLS:D=0:GOTO 160
270 IF C=1 AND D=1 THEN PRINT"N
AO HA ESPACOS DENTRO DE UMA
UNICA PALAVRA! TENTE DE NOVO":G
OSUB 950:CLS:D=0:GOTO 160
280 NEXT N
290 IF D<>C-1 THEN PRINT "VOCE
PRETENDE INTRODUZIR";C;"PALA-VR
AS ";I$:PRINT"TENTE DE NOVO":GO
SUB 950:CLS:D=0:GOTO 160
300 Z$=""
310 FOR N=1 TO L:Z$=Z$+"*":NEXT
 N
320 CLS:PRINT"SCORE/";A$,"SCORE
/";B$:PRINT @38,TA;TAB(22);TB;"
"
330 PRINT @69," VALOR DOS CARAC
TERES"
340 FOR N=0 TO 26:READ G$:Q$=Q$
+G$:NEXT N:PRINT Q$:RESTORE
350 PRINT @320,"A ";R$;" CONTEM
";L;"LETRAS":PRINT Z$
360 PRINT @416,"";:LINE INPUT "
XX=ADIVINHAR LETRA
 ZZ=ADIVINHAR A FRASE
  A-Z=COMPRAR O CARACTER   ?";D
$
900 DATA "A-20    ","B-10    ",
"C-10    ","D-12    ","E-20
","F-08    ","G-12    ","H-10
"
910 DATA"I-20    ","J-04    ","
K-06    ","L-10    ","M-10    "
,"N-10    ","O-20    ","P-10
","Q-02    ","R-12    ","S-12
"
920 DATA "T-12    ","U-20    ",
"V-08    ","W-08    ","X-04
","Y-08    ","Z-02    ","e-20
"
```

```
5 CLEAR 2000:KEYOFF:COLOR 15,4,
■
10 R$="palavra":W=14:F=1:TA=200
·TB=TA
40 CLS:LINEINPUT"Nome do jogado
r 1 (max 9 letras) ?      ";A$
50 PRINT·LINE INPUT"Nome do jog
ador 2 (max 9 letras) ?      ";B
$
60 IF LEN(A$)>9 OR LEN(B$)>9 TH
EN 40
70 CLS:PRINT"Qual o nível de di
ficuldade":INPUT"(número de pal
avras 1-9)";C$
90 IF C$<"1" OR C$>"9" THEN 70
100 C=VAL(C$)
110 PRINT:INPUT"Número de jogad
as (1-9)";T$
130 IF T$<"1" OR T$>"9" THEN 11
0
140 T=VAL(T$)
150 IF C>1 THEN J$="s"·I$="com
um espaço entre elas":R$="FRASE
"
160 CLS:PRINTA$;", é a sua vez.
```



```
":PRINT"Digite sua frase de";C;
"palavra";J$;"."
165 PRINT:PRINT"Se desejar ver
as letras digitadas, pressione
<0>, senão pressione <1>.";
170 K$=INKEY$:IF K$="" THEN 170
175 IF K$<>"1" AND K$<>"0" THEN
 170
180 IF K$="0" THEN 200 ELSE PRI
NT:PRINT"? ";
190 K$=INKEY$:IF K$="" THEN 190
195 IF K$<>CHR$(13) THEN S$=S$+
K$:GOTO 190 ELSE 220
200 PRINT:LINEINPUT"? ";S$
220 L=LEN(S$):PRINT
230 IF L=0 THEN PRINT"Entrada i
legal - repita":GOSUB 950:CLS:G
OTO 160
240 IF L>78 THEN PRINT"Entrada
muito longa - repita":GOSUB 950
:CLS:GOTO 160
250 FOR N=1 TO L:IF MID$(S$,N,1
)=CHR$(32) THEN D=D+1:GOTO 270
260 IF MID$(S$,N,1)<"A" OR MID$
(S$,N,1)>"Z"ANDMID$(S$,N,1)<>"Ç
"THEN PRINT"Caracter ilegal - r
epita":GOSUB 950:CLS:D=0:GOTO 1
60
270 IF C=1 AND D=1 THEN PRINT"E
spaços não são permitidos em um
a     única palavra - repita":
GOSUB 950:CLS:D=0:GOTO 160
280 NEXT
290 IF D<>C-1 THEN PRINT"Você d
eve digitar";C;"palavras ";I$;"
 - repita":GOSUB 950:CLS:D=0:GO
TO 160
300 Z$=""
310 Z$=STRING$(L,"*")
320 CLS:PRINTA$;TAB(25)B$:PRINT
TA;"pontos";TAB(25);TB;"pontos"
330 LOCATE 7,3:PRINT"Valores do
s caracteres"
340 Q$="":FOR N=1 TO 28:READ G$
:Q$=Q$+G$+STRING$(4+(N/5-INT(N/
5)),32):NEXT:PRINTQ$:RESTORE
350 LOCATE 0,11:PRINT"A ";R$;"
contém";L;"letras":PRINTZ$
360 LOCATE 0,21:PRINTSPC(77):LO
CATE 0,21:PRINT"XX-Adivinha let
ra     ZZ-Adivinha frase":INPUT"
            A-Z Compra letra ";D$
900 DATA A-20,B-08,C-12,Ç-06,D-
20,E-20,F-12,G-08,H-08,I-16,J-0
8,K-02,L-10,M-12,N-12
910 DATA O-20,P-08,Q-08,R-12,S-
20,T 12,U-16,V-08,W-02,X-08,Y-0
2,Z-06,_-20
```



```
10 R$ = "PALAVRA":W = 14:F = 1:
TA = 200:TB = TA
40  HOME : INPUT "NOME DO JOGAD
OR 1 (MAX 9 LET) ";A$
50  PRINT : INPUT "NOME DO JOGA
DOR 2 (MAX 9 LET) ";B$
60  IF  LEN (A$) > 9 OR  LEN (B
$) > 9 THEN 40
70  HOME : PRINT "NIVEL DE DIFI
CULDADE": INPUT "(NUMERO DE PAL
AVRAS 1-9)? ";C$
90  IF C$ < "1" OR C$ > "9" THE
N 70
100 C =  VAL (C$)
110  PRINT : INPUT "NUMERO DE J
OGADAS (1-9)? ";T$
130  IF T$ < "1" OR T$ > "9" TH
EN 110
140 T =  VAL (T$)
150  IF C > 1 THEN J$ = "S":I$
= "COM UM ESPACO ENTRE ELAS":R$
 = "FRASE"
160  HOME : PRINT A$", E A SUA
VEZ.": PRINT "DIGITE SUA FRASE
DE ";C;" PALAVRA";J$;"."
165  PRINT : PRINT "SE DESEJAR
VER AS LETRAS DIGITADAS, PRESSI
ONE <0>, SENAO PRESSIONE <1>.";

170  GET K$: IF K$ <  > "1" AND
 K$ <  > "0" THEN 170
180 K =  VAL (K$): PRINT : PRIN
T
185 S$ = ""
190  GET K$: IF K = 0 THEN  PRI
NT K$;
200  IF K$ =  CHR$ (8) AND  LEN
 (S$) > 1 THEN S$ =  LEFT$ (S$,
 LEN (S$) - 1): GOTO 190
210 S$ = S$ + K$: IF K$ <  >  C
HR$ (13) THEN 190
220 S$ =  MID$ (S$,2, LEN (S$)
- 2):L =  LEN (S$)
230  IF L = 0 THEN  PRINT : PRI
NT "ENTRADA ILEGAL - REPITA"; C
HR$ (7);: GOSUB 950: HOME : GOT
O 160
240  IF L > 40 THEN  PRINT : PR
INT "ENTRADA MUITO LONGA - REPI
TA"; CHR$ (7);: GOSUB 950: HOME
 : GOTO 160
250  FOR N = 1 TO L: IF  MID$ (
S$,N,1) =  CHR$ (32) THEN D = D
 + 1: GOTO 270
260  IF  MID$ (S$,N,1) < "A" OR
  MID$ (S$,N,1) > "Z" THEN  PRI
NT : PRINT "CARATER ILEGAL - RE
PITA": GOSUB 950: HOME :D = 0:
GOTO 160
270  IF C = 1 AND D = 1 THEN  P
RINT : PRINT "ESPACOS NAO SAO P
ERMITIDOS EM UMA        UNICA PAL
AVRA! - REPITA": GOSUB 950: HOM
E :D = 0: GOTO 160
280  NEXT
290  IF D <  > C - 1 THEN  PRIN
T : PRINT "VOCE DEVE ENTRAR ";C
;" PALAVRAS ";I$;" - REPITA": G
OSUB 950: HOME :D = 0: GOTO 160

300 Z$ = ""
310  FOR N = 1 TO L:Z$ = Z$ + "
*": NEXT
320  HOME : PRINT A$; TAB( 25).
B$: PRINT TA;" PONTOS"; TAB( 25
);TB;" PONTOS"
330  VTAB 4: HTAB 8: PRINT "VAL
ORES DOS CARACTERES";
340 Q$ = " ": FOR N = 1 TO 28:
READ G$:Q$ = Q$ + G$ + "     ":
NEXT : HTAB 40: PRINT Q$: RESTO
RE
350  VTAB 12: PRINT "A ";R$;" C
ONTEM ";L;" LETRAS";: HTAB 40:
PRINT Z$
360  VTAB 22: CALL  - 958: VTAB
 22: PRINT "XX-ADIVINHA LETRA
  ZZ-ADIVINHA FRASE": INPUT "
      A-Z COMPRA A LETRA ";D$
900  DATA  A-20,B-08,C-12,D-20
,E-20,F-12,G-08,H-08
910  DATA  I-16,J-08,K-02,L-10
,M-12,N-12,O-20,P-08,Q-08,R-12,
S-20
920  DATA  T-12,U-16,V-08,W-02,
X-04,Y-02,"     ",Z-06,_-20
```

O programa é muito parecido para todos os microcomputadores, já que não há gráficos que requeiram comandos especiais para montá-los. Apenas o início difere um pouco em alguns casos. No MSX e no TRS-Color é preciso reservar espaço para as variáveis alfanuméricas que serão usadas.

A linha 10 define todas as variáveis necessárias para o jogo. No TRS-Color, os **PEEK** da linha 15 são utilizados mais tarde para evitar que algumas letras apareçam na tela. As linhas 20 e 30 do Spectrum definem um UDG em branco para a tabela de letras.

As linhas 40 e 70 dão as mensagens iniciais do jogo. O nome dos jogadores é chamado pelas linhas 40 e 50. A linha 60 verifica se eles não são muito compridos para o espaço disponível na tela.



É possível transformar o programa em um jogo de palavras cruzadas?

Embora nosso programa não possa ser transformado num jogo desse tipo, ele certamente mostra o caminho para que o leitor elabore seu próprio programa.

Num jogo de palavras cruzadas, o jogador também deve descobrir quais são as letras de palavras "escondidas". A diferença é que no nosso jogo precisamos descobrir os caracteres de um cordão enquanto no de palavras cruzadas, buscamos elementos de uma matriz alfanumérica.

O tipo de pista que se dá ao jogador também é diferente. As instruções de um jogo de palavras cruzadas podem ocupar um espaço tão grande quanto os textos de um pequeno jogo de aventura. A disposição espacial das letras na tela também é muito importante. No programa, devemos correlacionar as coordenadas X e Y da tela aos números de linha da matriz de letras.



O número de palavras da frase é escolhido na linha 70.

As linhas 80 a 100 fazem verificações com a finalidade de se certificar de que o número de palavras por frase está dentro dos limites do programa.

Esse número é representado pela variável **C$**. A linha 80 (quando existente) verifica se a entrada é de apenas um caractere; a linha 90 checa se o valor digitado está entre os números 1 e 9. A linha 100 converte esse valor para uma variável numérica.

As linhas 110 a 140 estão relacionadas ao número de jogadas escolhido. A linha 110 apresenta a mensagem e obtém a resposta (variável **T$**). As linhas 120 e 130 são similares às linhas 80 e 90 A linha 140 converte o valor para uma variável numérica.

Se a frase for constituída de mais de uma palavra, a linha 150 diz ao jogador que coloque apenas um espaço entre cada palavra. **R$** é definida como "FRASE" para uso posterior.

O programa passa, então, para a rotina de entrada da frase que um dos jogadores deverá adivinhar. Ela vai da linha 160 até a 220 e fornece instruções para o jogador que vai digitar a frase. Se ele selecionar 0, a frase aparecerá na tela; caso contrário, ela permanecerá invisível. A frase é armazenada na variável **S$**.

As linhas 230 a 290 conferem a frase para verificar se está de acordo com as regras. Se ela não tiver nenhum caractere, isto é, se a tecla **<ENTER>** foi pressionada antes de se digitar qualquer letra, a linha 230 anuncia uma entrada ilegal e pede que se repita a operação. A linha 240 verifica o tamanho da frase e a 250 o número de espaços (que deve ser um a menos que o número de palavras).

As linhas 260 e 270 procuram caracteres ilegais. Observe que apenas letras maiúsculas serão aceitas, sem acento. No MSX, pode-se usar o "Ç".

As linhas 300 e 310 são encarregadas de definir a variável **Z$**, que contém a seqüência de asteriscos de número igual ao de letras de **S$**.

A rotina final, das linhas 320 a 360, monta o que falta da tela principal, lendo a tabela de valores de linhas **DATA** e colocando a tabela na posição adequada. A sequência de asteriscos também é posicionada. A linha 360 apresenta as opções para o jogador que vai adivinhar a frase.

APERFEIÇOE SEU BANCO DE DADOS

Apresentamos aqui novas rotinas para o seu banco de dados. Elas tornarão o programa mais eficiente, permitindo-lhe organizar melhor as informações que desejar.



- ADIÇÃO DE NOVAS OPÇÕES
- COMO MELHORAR AS ROTINAS JÁ EXISTENTES
- IMPRESSÃO CONTÍNUA
- UMA NOVA ORGANIZAÇÃO DO ARQUIVO
- BUSCA DE MÚLTIPLAS INFORMAÇÕES
- O USO DO CONTROLADOR DE DISCOS

O programa de banco de dados que desenvolvemos nos artigos das páginas 68 e 81 é muito útil para armazenar informações. Não importa que sejam detalhes de seu passatempo predileto, resultados de uma pesquisa, endereços de amigos e clientes ou qualquer outro tipo de dado. Devidamente arquivados, eles poderão ser consultados, corrigidos, alterados, apagados e mesmo impressos em papel.

Como qualquer outro programa de banco de dados, o nosso também não é perfeito para todas as aplicações possíveis. Para que você possa adaptá-lo da melhor maneira às suas necessidades, apresentamos algumas novas rotinas assim como sugestões para o aperfeiçoamento das já existentes. As rotinas, diferentes para cada computador, serão examinadas separadamente.

S

Para os usuários do Sinclair Spectrum, selecionamos duas rotinas: uma para a entrada contínua de registros e outra para impressão contínua.

ENTRADA CONTÍNUA

Estas linhas lhe permitirão dar entrada aos registros continuamente, sem precisar retornar ao menu. Elas impedem também que um cordão nulo — que provocaria a volta ao menu — seja aceito. Pressione <ENTER> quando completar a entrada dos dados.

```
2000 CLS : LET C=V
2110 FOR N=V TO A: PRINT  INVER
SE V;AT V+N*2,0;N$(N);AT V+N*2,
12; FLASH V;"?": INPUT "(ate ";
A(N);" caracteres)", LINE A$(C,
B(N)+V TO B(N+V)): IF N=V AND A
$(C,B(N)+V)=CHR$ 32 THEN  RETUR
■
2115 PRINT AT V+N*2,12;A$(C,B(N
)+V TO B(N+V)): NEXT N
2120 FOR F=V TO 150: NEXT F: IF
 C=V THEN  GOTO 2000
2140 IF A$(C)>=A$(C-V) THEN  GO
TO 2000
2150 LET X$=A$(C): LET A$(C)=A$
(C-V): LET A$(C-V)=X$: LET C=C-
V: IF C=V THEN  GOTO 2000
```

IMPRESSÃO CONTÍNUA

Como está, o programa possibilita a impressão apenas do registro em exame. Para imprimir a lista inteira, será necessário passar por todo o arquivo e comandar a impressão. Esta nova rotina faz o trabalho por você, iniciando pelo registro que está sendo exibido e imprimindo todos os seguintes até o fim da lista. Se quiser imprimir todos os registros, certifique-se de que está no registro 1, antes de iniciar o processo. Para interromper a listagem, pressione qualquer tecla.

```
 120 LET OP=1
3015 IF D-V=R THEN  LET D=D-V:
IF OP=6 THEN  LET OP=1
3020 IF A$(D,V)=CHR$ 32 THEN  L
ET D=D-V: IF OP=6 THEN  LET OP=
1
3085 IF OP=6 THEN  LET D=D+V: G
OTO 3010
4060 IF D>R THEN  LET D=PM: IF
OP=6 THEN  LET OP=1
4080 IF A$(D,V)=CHR$ 32 THEN  L
ET D=PM: IF OP=6 THEN  LET OP=1
4165 IF OP=6 THEN  LET MO=V: LE
T D=D+MO: GOTO 4060
9502 IF OP=6 AND INKEY$="" THEN
  COPY : RETURN
9505 PRINT  INVERSE V;AT 19,U;"
 impressao (C)ontinua";TAB 31;"
"
9585 IF V$="C" THEN  COPY : LET
 OP=6
```



T

Apresentamos quatro rotinas para o seu TRS-Color: impressão contínua, reordenação do arquivo, procura por múltiplos campos e adaptação para acionadores de disco.

IMPRESSÃO CONTÍNUA

Com esta rotina, você poderá imprimir todos os registros de uma só vez. Selecione a opção de impressão e escolha entre impressão contínua ou registro único. A impressão começa no registro que está sendo exibido na tela e vai até o fim da lista.

Se você quiser imprimir todo o arquivo, não se esqueça que deve começar com o registro 1.

```
1050 PRINT @385,"NUMERO DE CAMP
OS (1-8) ?";
1060 IN$=INKEY$:IF IN$<"1" OR I
N$>"8" THEN 1060
1070 A=VAL(IN$):DIM A(A),N$(A)
5070 IF D>NR AND G=1 THEN G=0:C
H=-1:CP=0 ELSE IF D>NR THEN CP=
0:GOTO 5230
5105 IF CP=1 GOSUB 10040:GOTO 5
160
5210 GOSUB 10000:GOTO 5160
6025 IF CP=1 GOSUB 10040:GOTO 6
080
6130 GOSUB 10000: IF CP=1 THEN
6080 ELSE 6030
6140 IF D>NR THEN D=1:CP=0
10000 PRINT @449," AJUSTE A IMP
RESSORA     CONT";STRING$(36,32
);
10010 IF INKEY$<>"C" THEN 10010
10020 PRINT @448,"CONTINUA OU S
OMENTE UM REGISTRO?";
10030 IN$=INKEY$:IF IN$<>"C" AN
D IN$<>"S" THEN 10030
10035 IF IN$="C" THEN CP=1
```

REORDENAÇÃO DO ARQUIVO

O programa original faz a ordenação dos registros sempre pelo mesmo campo (o primeiro). A nova alternativa, que aparecerá com o numero 8 no menu, permite que o campo-chave para a ordenação seja mudado. Depois da mudança, o arquivo é reordenado e o campo-chave passa a ser o primeiro da lista.

```
105 PRINT @388,"8-REORGANIZAR C
AMPOS"
120 IN$=INKEY$:IF IN$<"1" OR IN
$>"8" THEN 120
150 ON IN GOSUB 1000,2000,6000,
5000,7000,8000,9000,11000
11000 IF A=1 THEN PRINT" NAO PO
SSO REORGANIZAR 1 CAMPO!":FOR K
=1 TO 5000:NEXT:RETURN
11010 PRINT "NUM.DO CAMPO","NOM
E"
11020 FOR N=1 TO A:PRINT N,N$(N
):NEXT
11030 PRINT:PRINT" DIGITE O NUM
ERO DO NOVO CAMPO   CHAVE (2 A"
;A;")";:INPUT NC
11040 NC=INT(NC):IF NC<2 OR NC>
A THEN CLS:GOTO 11010
11050 CLS:PRINT" TROCANDO E ORD
ENANDO CAMPOS"
11060 T$=N$(1):N$(1)=N$(NC):N$(
```

```
NC)=T$:T=A(1):A(1)=A(NC):A(NC)=
T
11070 FOR N=1 TO NR:T$=A$(N,1):
A$(N,1)=A$(N,NC):A$(N,NC)=T$:NE
XT
11080 FOR N=1 TO NR-1:K=N
11090 FOR J=N+1 TO NR
11100 IF A$(J,1)<A$(K,1) THEN K
=J
11110 NEXT:IF N<>K THEN FOR C=1
 TO A:T$=A$(K,C):A$(K,C)=A$(N,C
)=T$:NEXT
11120 NEXT:RETURN
```

PROCURA POR MÚLTIPLOS CAMPOS

Esta nova opção permite que você busque mais de uma informação por vez. Quando se seleciona a rotina de busca a partir do menu principal, deve-se responder, para cada campo, o que será procurado. Você pode procurar por quantos campos desejar. Se um deles não lhe interessar, simplesmente tecle <ENTER> e vá em frente.



Em seguida, o computador perguntará se os dados especificados deverão estar presentes em todos os registros ou não. Se sua resposta for SIM, apenas os registros que contenham todos os dados especificados serão mostrados. Caso você responda NÃO, qualquer registro que contenha pelo menos um dos dados será exibido. Se, por exemplo, você solicitar o nome MARIA para o campo 1 e CAMPINAS para o campo 3, apenas pessoas que se chamem Maria E residam em Campinas serão apresentadas caso você tenha respondido SIM à pergunta anterior. Caso contrário, todas as pessoas que se chamem Maria OU residam em Campinas serão listadas.

```
5000 PRINT @B,B$;"opcao";B$;"de
";B$;"procura";B$
5010 BT=0:PRINT:FOR N=1 TO A:PR
INT "PROCURA PELO QUE NO CAMPO"
:PRINT N;", ";N$(N);" ?"
5020 LINEINPUT S$(N):IF S$(N)<>
"" THEN BT=BT+1
5025 NEXT:IF BT=0 THEN RETURN
5027 IF BT<2 THEN BT=0:GOTO 506
0
5030 PRINT:PRINT" TODOS OS DADO
S DEVEM ESTAR SI-  MULTANEAMENT
E NO MESMO REGISTRO (S/N) ?";
5040 IN$=INKEY$:IF IN$<>"S" AND
 IN$<>"N" THEN 5040
5050 CLS:BT=0:IF IN$="S" THEN B
T=1
5090 FOR Z=1 TO A:PS=INSTR(A$(D
,Z),S$(Z)):IF PS>0 AND BT=0 AND
 S$(Z)<>"" THEN Z=A:NEXT:GOTO 5
100
5093 IF PS=0 AND BT=1 THEN Z=A:
NEXT:D=D+CH:GOTO 5070
5096 NEXT:IF BT=0 THEN D=D+CH:G
OTO 5070
5230 CLS 2:PRINT "  NENHUM REGI
STRO COM";:IF BT=1 THEN PRINT @
21," TODOS OS",ELSE PRINT @21,"
 ALGUM DOS"
5231 PRINT "  DADOS FOI ENCONTR
ADO";
5235 FOR Z=1 TO A:IF S$(Z)="" T
HEN 5245
5240 PRINT @96+Z*32,N$(Z);:PRIN
T @107+Z*32,S$(Z);
5245 NEXT:CP=0
```

O USO DO ACIONADOR DE DISCOS

As linhas que se seguem fazem com que o programa trabalhe com o auxílio de um acionador de discos. Se você usa um gravador cassete, não as digite, pois as rotinas de carregamento e gravação serão alteradas.

Antes de introduzir as novas linhas,

você deverá apagar as linhas 8060 a 8070, 8150 a 8200 e 7090 a 7140. A maneira mais fácil de fazê-lo é digitar **DEL nº inicial nº final.** Por exemplo: **DEL 8060-8070.**

Para transferir dados da fita cassete para o acionador de discos, carregue o programa e faça as alterações da rotina de gravação (linhas até 7080). Carregue os dados e grave-os em disco. Apague, então, as linhas de 8060 em diante (só as especificadas antes). Por fim, digite as novas linhas da rotina de carregamento de dados.

```
80 PRINT@292,"5-SALVAR ARQUIVO"
90 PRINT @324,"6-CARREGAR ARQUI
VO"
1130 NEXT:R=INT(9000/(5+5*A))-1
:PRINT "NUMERO MAXIMO DE REGIST
ROS=";R
1140 DIM A$(R,A):FOR I=1 TO 200
0:NEXT:RETURN
7000 CLS:PRINT" CERTIFIQUE-SE D
E QUE O DRIVE     ESTA LIGADO E
O DISCO INSERIDO  E PRESSIONE <
ENTER>
7010 IF INKEY$<>CHR$(13) THEN 7
010
7020 PRINT:PRINT" QUAL O NOME D
O ARQUIVO ?";:LINEINPUT FI$
7030 IF LEFT$(FI$,1)<"A" OR LEF
T$(FI$,1)>"Z" THEN 7020
7040 OPEN "O",#1,FI$+"/DAT":CLS
6:PRINT @232," GRAVANDO ";FI$;
7050 WRITE #1,R,A,NR
7060 FOR N=1 TO A:WRITE #1,N$(N
),A(N):NEXT
7070 FOR C=1 TO NR:FOR N=1 TO A
:WRITE #1,A$(C,N):NEXT N,C
7080 CLOSE#1:RETURN
8030 PRINT @65,"SELECIONE O DIS
CO, PRESSIONE     <ENTER>"
8040 IF INKEY$<>CHR$(13) THEN 8
040
8050 IF R>0 THEN RUN 9210
8080 PRINT:PRINT" QUAL O NOME D
O ARQUIVO ?";:LINEINPUT FI$
8090 IF LEFT$(FI$,1)<"A" OR LEF
T$(FI$,1)>"Z" THEN 8080
8100 OPEN "I",#1,FI$+"/DAT"
8105 INPUT#1,R,A,NR
8110 DIM A(A),N$(A),A$(R,A)
8120 FOR N=1 TO A:INPUT#1,N$(N)
,A(N):NEXT
8130 FOR C=1 TO NR:FOR N=1 TO A
:INPUT#1,A$(C,N):NEXT N,C
8140 CLOSE#1:RETURN
```

IMPRESSÃO CONTÍNUA

Apresentamos três rotinas a mais para o MSX. Com elas você poderá fazer uma listagem contínua dos seus dados, reorganizar o arquivo e buscar várias informações ao mesmo tempo.

A rotina para impressão contínua permite que vários registros do seu arquivo sejam listados de uma só vez. O registro a partir do qual você quer começar a listagem deve estar sendo exibido na tela. Selecione, então, a opção [I]mprimir e, a seguir, a opção [C]ontínua. Todos os registros a partir deste serão listados.

Com a opção [R]egistro você pode imprimir apenas aquele que está sendo mostrado. Não se esqueça de que, para listar todo o arquivo, você deve ter o registro nº 1 da tela.

```
5070 IFD>NRANDG=1THEN G=0:CH=-1
:CP=0 ELSEIFD>NRTHEN CP=0 :GOTO
 5230
5105 IF CP=1 THEN GOSUB 10040:G
OTO 6080
5210 GOSUB10000:GOTO5160
6025 IF CP=1 THEN GOSUB 10040:G
OTO 6080
6130 GOSUB10000:IF CP=1 THEN 60
80 ELSE 6020
6140 IFD>NRTHEND=1:CP=0
10030 PRINT:LOCATE 1:PRINT"IMPR
ESSAO [C]ONTINUA OU [R]EGISTRO?
 ";
10035 IN$=INKEY$:IFIN$<>"C"ANDI
N$<>"R"THEN10035
10037 IFIN$="C" THEN CP=1
```

REORGANIZAÇÃO DO ARQUIVO

Esta nova opção permite que se troque o campo-chave para a ordenação do arquivo. O programa original tomava automaticamente o primeiro campo para fazer a ordenação alfanumérica dos registros. Com a possibilidade de trocar esse campo, ficará fácil, por exemplo, organizar por autor um arquivo de livros que estava indexado por título da obra. A operação pode ser revertida a qualquer momento. O novo campo-chave aparecerá em primeiro lugar na listagem dos campos.

```
105 LOCATE 9,19:PRINT"8:-REORGA
NIZAR O ARQUIVO"
110 LOCATE 14,22:PRINT"OPÇAO: "
;
120 IN$=INKEY$:IF IN$<"1"ORIN$>
"8"THEN120
150 ON IN GOSUB 1000,2000,6000,
5000,7000,8000,9000,11000
11000 IF A=1 THEN PRINT:PRINT"I
MPOSSIVEL REORGANIZAR APENAS 1
CAMPO!":FOR K=1 TO 5000:NEXT:RE
TURN
11010 PRINT:PRINT"CAMPO N.","NO
ME":PRINT
11020 FOR N=1 TO A:PRINTN,N$(N)
:NEXT
11030 PRINT:PRINT"DIGITE O NUME
RO DO NOVO CAMPO CHAVE":PRINT"(
 2 A";A;")";:INPUT NC
11040 IF NC<2 OR NC>A THEN CLS:
GOTO 10010
11050 CLS:LOCATE7,18:PRINT"REOR
GANIZANDO E REORDENANDO":PRINT:
PRINTTAB(15)"O ARQUIVO"
11060 SWAP N$(1),N$(NC):SWAP A(
1),A(NC)
11070 FOR N=1 TO NR:SWAP A$(N,1
),A$(N,NC):NEXT
11080 FOR N=1 TO NR-1:K=N
11090 FOR J=N+1 TO NR
11100 IF A$(J,1)<A$(K,1) THEN K
=J
11110 NEXT:IF N<>K THEN FOR C=1
 TO A:SWAP A$(K,C) A$(N,C):NEXT
11120 NEXT:RETURN
```



Como modificar o programa para trabalhar com arquivos em disco?

Os micros das linhas TRS-Color e MSX podem usar discos flexíveis como meio magnético de armazenamento de dados. Os programas de INPUT, contudo, destinam-se a equipamentos que utilizam fitas cassete.

Se quisermos usar em discos nosso programa para gerar e controlar arquivos, precisaremos modificar as seções que abrem estes arquivos e as que gravam e recuperam dados deles. O procedimento é necessário porque o programa não modifica registros isolados no arquivo gravado, e sim na memória do computador. Um conjunto de registros é criado na memória e, depois, gravado na fita. Quando queremos alterar alguma coisa, todo o bloco é trazido para a memória, onde o programa faz a edição. Depois que o processo se completa, gravamos o bloco em fita.

Para fazer essas modificações, o usuário precisa, naturalmente, conhecer os comandos correspondentes para manipulação de arquivos em disco. Estes deverão ser do tipo seqüencial, como os da fita.

BUSCA DE MÚLTIPLAS INFORMAÇÕES

Com a rotina dada a seguir você terá a alternativa de procurar por mais de uma informação de cada vez. Quando selecionar a opção de busca, você precisará especificar as informações que deseja em cada campo do registro. Se um determinado campo não lhe interessa, simplesmente tecle **<RETURN>**. Depois, o computador perguntará se os dados especificados devem estar simultaneamente em cada registro ou não. A resposta afirmativa corresponde a um E na

busca — ou seja, o registro deve conter isto E isto E aquilo. O contrário corresponde a um OU. Vejamos um exemplo. Você procura por MACHADO DE ASSIS no campo AUTOR e ROMANCE no campo GÊNERO LITERÁRIO. Se sua resposta para a pergunta anterior for SIM, apenas os romances de Machado de Assis serão listados. Se sua resposta for NÃO, todos os livros de Machado e todos os romances serão listados.

```
5000 PRINT"BUSCA DE INFORMACOES
"
5010 BT=0·PRINT:FOR N=1 TO A:PR
INT"PROCURAR O QUE EM ";N$(N);"
? ";:S$(N)=""
5020 LINEINPUTS$(N)·IF S$(N)<>"
" THEN BT=BT+1
5025 NEXT·IF BT=0 THEN RETURN
5027 IF BT=1 THEN BT=0:GOTO 506
0
5030 PRINT·PRINT·PRINT"TODOS OS
 DADOS DEVEM ESTAR PRESENTES
 SIMULTANEAMENTE? ";
5040 IN$=INKEY$·IF IN$<>"S" AND
 IN$<>"N" THEN 5040
5050 CLS:BT=0:IF IN$="S" THEN B
T=1
5090 FOR Z=1 TO A·PS=INSTR(A$(D
,Z),S$(Z))·IF PS>0 AND BT=0 AND
 S$(Z)<>"" THEN Z=A·NEXT·GOTO 5
100
5093 IF PS=0 AND BT=1 THEN Z=A:
NEXT·D=D+CH·GOTO 5070
5096 NEXT·IF BT=0 THEN D=D+CH·G
OTO 5070
5230 CLS LOCATE 5,10 PRINT"NENH
UM REGISTRO COM ";·IF BT 1 THEN
 PRINT"TODOS OS" ELSE PRINT"ALG
UM DOS"
5235 PRINT"FOI ENCONTRADO!":CP=
0
```



Para o Apple II, temos quatro rotinas. As três primeiras possibilitam a impressão contínua de registros, a reorganização do seu arquivo e a busca de várias informações ao mesmo tempo. A última delas melhora rotinas já existentes no programa, tornando-o, desse modo, mais completo.

IMPRESSÃO CONTÍNUA

Esta pequena rotina permite que os registros sejam listados em sequência, e não apenas um de cada vez. Deixe na tela o primeiro registro a ser impresso e escolha a opção de impressão. Em seguida, tecle C para a impressão contínua. Todos os registros, a partir do que está na tela, serão listados. Para a listagem de todo o arquivo, deixe o registro 1 na tela e proceda como foi explicado.

```
10017  VTAB 23· HTAB 5· CALL -
 958: PRINT "IMPRESSAO ";: INVE
RSE · PRINT "C";· NORMAL : PRIN
T "ONTINUA OU ";· INVERSE · PRI
NT "R";: NORMAL · PRINT "EGISTR
O ?";
10018  GET IN$· IF IN$ <  > "C"
 AND IN$ <  > "R" THEN 10018
10025  IF IN$ = "C" THEN E = D·
 FOR D = E TO NR
10052  IF IN$ = "C" THEN  NEXT
:D = E
```

REORGANIZAÇÃO DO ARQUIVO

O programa original usava o primeiro campo do arquivo como campo-chave para a ordenação dos registros. Com esta rotina, você poderá trocar o campo. Selecione a opção 8 a partir do menu principal e especifique qual o novo campo-chave. O arquivo será, então, reordenado de acordo com a escolha feita. O novo campo-chave aparecerá no topo da lista de campos.

```
105  PRINT · HTAB 10· PRINT "8·
-REORGANIZAR O ARQUIVO"
110  VTAB 23· HTAB 15: PRINT "O
PCAO: ";
130  IF IN$ < "1" OR IN$ > "8"
THEN 110
150  ON IN GOSUB 1000,2000,6000
,5000,7000,8000,6520,11500


11500  IF A = 1 THEN  VTAB 10:
HTAB 9· PRINT "IMPOSSIVEL REORG
ANIZAR"· PRINT  TAB( 13)"APENAS
 1 CAMPO!": FOR K = 1 TO 3000:
NEXT : RETURN
11510  PRINT · PRINT "CAMPO NO.
","NOME"
11520  FOR Z = 1 TO A· PRINT Z,
N$(Z)· NEXT
11530  PRINT : PRINT "DIGITE O
NUMERO DO NOVO CAMPO CHAVE"· PR
INT "(DE 2 A ";A;")";: INPUT NC
%
11540  IF (NC% < 2) OR (NC% > A
) THEN  HOME : GOTO 11510
11550  HOME  VTAB 10: HTAB 8
PRINT "AGUARDE A  REORGANIZACAO
": PRINT  TAB( 9)"E ORDENACAO D
O ARQUIVO"
11560 T$ = N$(1):N$(1) = N$(NC%
):N$(NC%) = T$:T = A(1):A(1) =
A(NC%):A(NC%) = T
11570  FOR N = 1 TO NR:T$ = A$(
N,1):A$(N,1) = A$(N,NC%):A$(N,N
C%) = T$: NEXT
11580  FOR N = 1 TO NR - 1:K =
N
11590  FOR J = N + 1 TO NR
11600  IF A$(J,1) < A$(K,1) THE
N K = J
11610  NEXT : IF N <  > K THEN
 FOR C = 1 TO A:T$ = A$(K,C)·A$
(K,C) = A$(N,C):A$(N,C) = T$: N
EXT
11620  NEXT : RETURN
```

BUSCA DE MÚLTIPLAS INFORMAÇÕES

Esta é uma rotina destinada a agilizar a consulta ao seu arquivo de dados, permitindo-lhe procurar por mais de uma informação de cada vez. Quando você solicitar a opção de busca, precisará especificar as informações que deseja em cada campo. Se um campo não lhe interessa, simplesmente tecle <ENTER>. Depois, o computador perguntará se os dados especificados devem estar simultaneamente em cada registro ou não. A resposta afirmativa fará com que apenas os registros que contêm todos os dados especificados sejam apresentados. A resposta negativa levará todo registro que contenha ao menos um dos dados a ser listado.

```
5000 DD = 1:BT = 0: PRINT : FOR
 X = 1 TO A· PRINT : PRINT "PRO
CURAR O QUE EM ";N$(X);
5010  INPUT S$(X): IF S$(X) <
> "" THEN BT = BT + 1
5020  NEXT : IF BT = 0 THEN  RE
TURN
5025  IF BT ÷ 1 THEN BT = 0· GO
TO 5040
5030  VTAB 21· HTAB 1· PRINT "T
ODOS OS DADOS DEVEM ESTAR PRESE
NTES    EM CADA REGISTRO? (S/N)
";
5035  GET IN$· IF IN$ <  > "S"
AND IN$ <  > "N" THEN 5035
5037  IF IN$ = "N" THEN BT = 0
5040  VTAB 23: HTAB 13: PRINT "
PROCURANDO...";
5050  FOR XX = DD TO NR
5052  FOR Z = 1 TO A: IF S$(Z)
= "" THEN 5060
5054 PS =  (S$(Z) =  LEFT$ (A$(X
X,Z), LEN (S$(Z))))· IF PS AND
( NOT BT) THEN Z = A: NEXT : GO
TO 5065
5056  IF ( NOT PS) AND BT THEN
Z = A: NEXT · NEXT · GOTO 5070
5060  NEXT : IF BT = 0 THEN  NE
XT : GOTO 5070
5065 FL = 1:D = XX· GOSUB 6020:
 GOTO 5200
5070  IF FL = 0 THEN  VTAB 21·
HTAB 1: CALL  - 958: PRINT "NAO
 ENCONTREI NENHUM REGISTRO COM
OS      DADOS SOLICITADOS!": FOR
 X = 1 TO 5000: NEXT : GOTO 509
0
5090 FL = 0: RETURN
5200  IF XX = NR THEN 5070
5210  VTAB 15· PRINT "CONTINUO
PROCURANDO? (S/N) ";· GET IN$
```

MELHORE AS ROTINAS EXISTENTES

Estas linhas tornam seu programa de banco de dados mais completo. Com elas você poderá especificar o *slot* e a unidade de disco que usará para a gravação e o carregamento dos dados. A opção padrão é slot 6 e *drive* 1. Um segundo drive poderá ser acionado diretamente do programa.

Alteramos ainda a rotina de detecção de erros. Ela está mais explícita em relação à não existência de um arquivo especificado para a leitura de dados. E também mais equipada para prevenir que qualquer outro erro (a seleção de um drive que não existe, por exemplo) interrompa seu programa e provoque a perda de dados.

```
22 SS = 6:DR = 1
25  ONERR  GOTO 11000
1030  IF R > 0 THEN  CLEAR ·D$
=  CHR$ (4):IN = 1:SS = 6:DR =
1· HOME : GOTO 150
1050  VTAB 3: HTAB 5: PRINT "NU
MERO DE CAMPOS (1-8): ";
1060  GET A$· IF A$ > "8" OR A$
 < "1" THEN 1060
1070  PRINT A$:A =  VAL (A$): D
IM A(A),N$(A)
3010  VTAB 23: CALL  - 958: PRI
NT "NUMERO DO CAMPO A SER MODIF
ICADO =>";
3020  GET CP$·CP =  VAL (CP$)
3030  IF CP > A OR CP < 1 THEN
 POKE 34,0: RETURN
7010  VTAB 15· HTAB 5: PRINT "S
LOT ";SS; CHR$ (8);· GET SS$  I
F SS$ =  CHR$ (13) THEN SS$ =
STR$ (SS)
7015  PRINT SS$:SS =  VAL (SS$)
: IF SS < 1 OR SS > 7 THEN 7010
7020  VTAB 17: HTAB 5· PRINT "D
RIVE ";DR; CHR$ (8);· GET DR$·
IF DR$ =  CHR$ (13) THEN DR$ =
 STR$ (DR)
7022  PRINT DR$·DR =  VAL (DR$)
: IF DR < 1 OR DR > 2 THEN 7020
7025  PRINT : PRINT D$,"OPEN",A
R$;",S";SS;",D";DR: PRINT D$;"D
ELETE";AR$
8020  HOME : CLEAR :D$ =  CHR$
(4):IN = 6:SS = 6·DR = 1
8050  VTAB 15: HTAB 5: PRINT "S
LOT ";SS; CHR$ (8);  GET SS$· I
F SS$ =  CHR$ (13) THEN SS$ =
STR$ (SS)
8055  PRINT SS$:SS =  VAL (SS$)
: IF SS < 1 OR SS > 7 THEN 8050
8060  VTAB 17· HTAB 5· PRINT "D
RIVE ";DR; CHR$ (8);: GET DR$·
IF DR$ =  CHR$ (13) THEN DR$ =
 STR$ (DR)
8065  PRINT DR$·DR =  VAL (DR$)
: IF DR < 1 OR DR > 2 THEN 8060
8070  PRINT · PRINT D$;"OPEN";A
R$;",S";SS;",D";DR
11000  IF  PEEK (222) <  > 5 TH
EN  VTAB 24: HTAB 13: PRINT "HO
UVE UM ERRO!";: FOR I = 1 TO 10
00: NEXT · GOTO 30
11010  PRINT D$;"CLOSE"
11020  VTAB 24· HTAB 8· PRINT "
ESTE ARQUIVO NAO EXISTE!";: FOR
 I = 1 TO 1000: NEXT : HOME : G
OTO 150
```

MICRO DICAS

NÃO PERCA SEUS DADOS

Às vezes, por descuido ou erro de digitação, o programa é interrompido por uma mensagem de erro. Se executarmos o programa de novo usando **RUN**, todas as variáveis criadas pelo programa serão apagadas.

A saída para o problema está em evitar o comando **RUN** e voltar a executar o programa de outro modo.

Em geral, é possível encontrar um ponto onde o programa recomeça como se nada tivesse acontecido. Na maioria das vezes, este ponto corresponde à parte que cuida do menu. Para executar o programa novamente, bastará, então, anotar o número da linha e usar o comando **GOTO**.

APPLE E TK-2000: EFEITOS SONOROS

O comando SOUND do TK-2000 tem certas limitações. Já o Apple não possui um comando BASIC para produzir sons. Veja como usar código de máquina para criar efeitos sonoros.

Os usuários do Apple ressentem-se, sem dúvida, da ausência de um comando em BASIC para produzir sons. Essa limitação pode ser contornada com o uso de rotinas em código de máquina.

Estalidos ou "cliques" produzidos no alto-falante constituem o máximo que se pode obter com um programa BASIC, no Apple, em matéria de som. Para isso (tanto no Apple quanto no TK-2000), basta que façamos referência à posição de memória -16336 por meio de um comando **PEEK**:

```
X = PEEK(-16336)
```

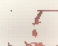
Quando testamos o valor desse endereço especial, o cone do alto-falante se move — para dentro e para fora — uma vez. Se quisermos produzir notas musicais, precisaremos imprimir a esse movimento uma velocidade que somente programas em linguagem de máquina permitem. Controlando a velocidade com que o cone do alto-falante se move para frente e para trás, poderemos controlar também a freqüência da nota musical ou do ruído emitido. Quanto mais rápido for o movimento do cone, mais agudo será o som.

COMO CONTROLAR O ALTO-FALANTE

A rotina em Assembly fornecida a seguir produz um som que vai se tornando cada vez mais agudo. Use nosso Assembler para montá-la na memória.



```
10 ORG 800
20 LDA #$00
30 STA $FF
40 LOOP LDA #$00
50 STA $C030
60 LDX $FF
70 PAUSE NOP
80 NOP
90 NOP
100 NOP
110 DEX
120 BNE PAUSE
130 DEC $FF
140 BEQ FIM
150 JMP LOOP
160 FIM RTS
170 END
```



Os usuários do TK-2000 podem recorrer ao míni-Assembler embutido no micro. O míni-Assembler, contudo, não aceita rótulos, seguindo outra listagem com os endereços já calculados. Ao contrário da listagem anterior, esta não é relocável na memória.

```
0320-   LDA   #$00
0322-   STA   $FF
0324-   LDA   #$00
0326-   STA   $C030
0329-   LDX   $FF
032B-   NOP
032C-   NOP
032D-   NOP
032E-   NOP
032F-   DEX
0330-   BNE   $032B
0332-   DEC   $FF
0334-   BEQ   $0339
0336-   JMP   $0324
0339-   RTS
```

Se você usar nosso Assembler para montar o programa, verá que ele emite

- COMO CONTROLAR O ALTO-FALANTE
- O USO DE CONTADORES
- O CONTROLE DA FREQÜÊNCIA DA NOTA
- COMO USAR LAÇOS VAZIOS PARA PRODUZIR PAUSAS DE DIVERSOS TAMANHOS
- UMA ROTINA DE SOM PARA O APPLE

mensagens de erro durante a montagem. Para corrigir esse defeito, modifique a linha 110: apague a segunda vírgula depois do **NOP** e acrescente uma vírgula após o número 234.

Qualquer referência à posição de memória -16336 em decimal, ou $C030 em hexadecimal, provocará um movimento no alto-falante. Essa referência pode ser feita por meio de comandos como **STA, INC** ou **DEC**. O importante, de fato, é controlar a freqüência com que isso ocorre no programa.

O USO DE CONTADORES

A posição de memória $FF — que fica na página zero — vai ser usada como contador. Para começar, colocamos nela o valor zero, pelos comandos **LDA #$00** e **STA $FF**. Depois, o comando **STA $C030** provoca um primeiro movimento do alto-falante. Note que, antes disso, foi colocado zero no registro A. Na realidade, o valor transferido de A para o endereço $C030 pelo comando **STA $C030** pode ser qualquer um: não faz a menor diferença.



A seguir, a instrução **LDX $FF** coloca no registro X o conteúdo da memória $00FF. Esse tipo de endereçamento é denominado *endereçamento direto em página zero*. Sua vantagem consiste em permitir a especificação de um endereço com apenas um byte. Como um byte pode conter um número entre 0 e 255, o endereço especificado deve ficar num dos primeiros 256 bytes da memória — a chamada *página um*.

As quatro instruções **NOP** (Nenhuma **OP**eração) utilizadas em seguida nada executam. Sua função é reduzir a velocidade do programa. A instrução **DEX** subtrai uma unidade do valor de X. Se o resultado da subtração não for igual a zero, a instrução **BNE** manda o processador de volta ao rótulo **PAUSE**. O laço que fica entre as linhas 70 e 110 é repetido, então, 256 vezes — lembre-se de que zero menos um é igual a 255 em linguagem de máquina.

Em seguida, a instrução **DEC $FF** subtrai uma unidade do conteúdo do endereço **$FF**. Se a operação resultar em zero, a instrução **BEQ FIM** envia o processador para o rótulo **FIM**, onde o comando **RTS** termina a rotina. Se a operação não resultar em zero, o programa segue seu curso natural e a instrução **JMP LOOP** envia o processador ao rótulo **LOOP**, para a produção de um novo movimento do alto-falante.

Assim, após a emissão do primeiro som, ocorre uma pausa equivalente a 256 voltas do laço. O conteúdo de $FF, que inicialmente é zero — equivalente a 256, em código de máquina —, controla a duração da pausa, diminuindo em uma unidade após esta. Enquanto não chegar a zero, um novo movimento do alto-falante será produzido. Como o conteúdo de $FF, que controla o tamanho da pausa, diminui a cada volta do laço entre as linhas 40 e 150, o tamanho da pausa entre dois movimentos do alto-falante também diminui. Dessa maneira, a freqüência do som produzido vai aumentando, ou seja, o som vai se tornando cada vez mais agudo.

No TK-2000, o som é produzido por intermédio do alto-falante da TV. Já no Apple um dispositivo sonoro que está

embutido no próprio computador emite o ruído — e a qualidade do som deixa a desejar.

A VEZ DO APPLE

A rotina de produção de ruídos que apresentamos a seguir pode ser utilizada a partir de programas BASIC, equivalendo ao comando **SOUND** dos micros da linha TK-2000.

Ela permite ao usuário do Apple produzir notas musicais em seus programas sem ter que programar uma rotina em linguagem de máquina para cada novo efeito sonoro.



```
10 ORG 800
20 LDY #$00
30 SOM LDX $385
40 INC $C030
50 PAUSET DEY
60 BNE PAUSES
70 DEC $384
80 BEQ FIM
90 PAUSES DEX
100 BNE PAUSET
110 JMP SOM
120 FIM RTS
```



Embora o TK-2000 dispense esse tipo de rotina, seus usuários podem querer montá-la para aprender ou, simplesmente, compará-la ao comando **SOUND**.

```
0320-   LDY    #$00
0322-   LDX    $0385
0325-   INC    $C030
0328-   DEY
0329-   BNE    $0330
032B-   DEC    $0384
032E-   BEQ    $0336
0330-   DEX
0331-   BNE    $0328
0333-   JMP    $0322
0336-   RTS
```

Após a definição do endereço inicial, o comando **LDY #$00** coloca zero no registro Y, que será utilizado como contador. Não se esqueça de que zero equivale a 256.

O endereço $385 conterá a tonalidade da nota emitida — em outras palavras, seu valor vai estabelecer o tamanho de um laço de pausa para controlar a freqüência do movimento do alto-falante. Assim, o conteúdo dessa posição é colocado em X, que também será usado como contador.

A instrução **INC $C030** produz um movimento do alto-falante. A seguir, o programa apresenta um laço complexo, controlado por três contadores. Estes estabelecem tanto a freqüência quanto a duração do som emitido. Vejamos, então, como o laço funciona.

Inicialmente, a instrução **DEY**, na linha 50, subtrai uma unidade do conteúdo de Y. A instrução seguinte — **BNE PAUSES** — faz com que o programa salte para a linha 90, enquanto o valor em Y não for reduzido a zero. A linha 90, por sua vez, contém uma instrução **DEX**, que diminui o conteúdo de X em uma unidade. De maneira análoga, essa instrução é seguida por um **BNE PAUSET**, que retorna à linha 50 enquanto o valor de X não chegar a zero. Assim, o processador fica "preso" em um laço, saindo apenas quando o conteúdo de X ou Y for zero.

Os contadores trabalham independentemente. Quando o valor de X se torna zero, o processador passa para a linha 110, que retorna ao rótulo **SOM (JMP SOM)**, onde o valor de $385 (tonalidade) é recolocado em X e se produz um novo movimento do alto-falante. Assim, o valor inicial de X determina o tamanho da pausa entre dois ruídos, independentemente do que acontece com Y.

O registro Y continha inicialmente o número 256 — ou zero. Portanto, após 256 passagens pela linha 50, o valor de Y se torna zero, e o desvio da linha 60 não ocorre. O programa, então, prossegue na linha 70 — **DEC $384** —, que diminui em uma unidade o valor em $384 (contador que controla a duração da nota). Assim, independentemente do que acontece com X, a cada 256 voltas do laço principal, o endereço $384 é diminuído. Quando ele se torna zero, a linha 80 — **BEQ FIM** — envia o processador para a linha 120, onde o comando **RTS** retorna ao BASIC. O valor inicialmente colocado em $0384 controla, dessa maneira, a duração da nota.

MICRO DICAS

O MINI-ASSEMBLER

O mini-Assembler é um programa Assembler simplificado, mas extremamente útil. Ele está disponível no TK-2000 bem como em microcomputadores Apple que tenham o INTEGER BASIC disponível em ROM ou numa placa de expansão de memória. Sua principal limitação é não aceitar rótulos.



Como usar o monitor-Disassembler?

Uma ferramenta muito útil aos usuários do Apple, e indispensável aos do TK-2000, é o monitor Disassembler. Embora não precisemos de monitor para montar os programas código por código, ele nos permitirá verificar se a montagem foi bem-sucedida.

Para entrar no monitor, digite:

```
CALL -151
```

ou **LM** no TK 2000.

Para listar um programa que comece no endereço 800, digite (após ter entrado no monitor):

```
320L
```

Agora você pode produzir sons em programas BASIC. Para ter uma idéia de como proceder, digite este exemplo:



```
10  FOR S = 1 TO 255
20  POKE 900,50
30  POKE 901,S
40  CALL 800
50  NEXT
```

O programa consiste em um laço, onde S tem a função de controlar a freqüência da nota — de fato, S é proporcional ao *período* da nota, que é o inverso da freqüência.

O endereço 900, que corresponde a $384 em hexadecimal, contém a duração da nota. A linha 20 estabelece esse parâmetro por meio de um **POKE**. A mesma instrução é usada na linha 30 para definir a tonalidade S da nota. O número 901 em decimal corresponde a $385 em hexadecimal.

Finalmente, a instrução **CALL 800** chama a rotina em código, produzindo notas cada vez mais graves à medida que as voltas do laço se sucedem.



Os usuários do TK-2000 podem comparar a rotina em código com o comando **SOUND** apagando as linhas de 20 a 40 e acrescentando:

```
30 SOUND S,50
```

DE OLHO NA TELA

- COMO DETECTAR UMA FIGURA
- OS COMANDOS **ATTR, SCRN, POINT, PPOINT**
- PROGRAMA DA BOLA ELÁSTICA
- COMO DETECTAR COLISÕES



Conheça os comandos que permitem ao computador "olhar" sua própria tela. Eles são muito úteis na manipulação de gráficos complicados e, sobretudo, em jogos onde ocorrem colisões.

Ao criar uma tela gráfica detalhada, como garantir que uma figura não coincida com outra já presente?

Uma alternativa seria tomar nota das áreas da tela que estão sendo ocupadas. Mas a tarefa, além de fatigante, muitas vezes é impossível — como no caso de gráficos em movimento. E esse tipo de problema se coloca sempre que programamos um jogo, onde, por exemplo, naves e alienígenas movem-se pela galáxia ou robôs tentam sair de um labirinto. Nos dois casos, precisamos nos assegurar de que duas figuras não serão desenhadas no mesmo lugar ou que algo especial acontecerá se elas colidirem — uma explosão, digamos.

COMO DETECTAR UMA FIGURA

Suponhamos que você queira escrever um programa no qual uma bola fique batendo nas bordas da tela. Não será difícil: conhecendo coordenadas dos quatro lados da tela, bastará incluir um teste de condição **IF...THEN** para verificar se a bola está ou não batendo na borda. Para isso, as coordenadas da bola são comparadas com as coordenadas já conhecidas das bordas. Mas o que aconteceria se quiséssemos checar se a bola bateu na borda de um objeto cuja forma é mais complicada — um círculo, por exemplo?

Poderíamos usar o mesmo método: um certo número de **IF...THEN**, contendo informações sobre as coordenadas do círculo, verificaria se a bola ba-

teu na borda deste. A forma curvilínea é relativamente mais complexa e, por isso, precisaríamos fazer muitos testes. Como sabemos, o computador demora para realizar um teste **IF...THEN**. Assim, um programa cheio deles se tornaria extremamente lento.

Existe um limite de velocidade de execução num programa em BASIC, mas o Spectrum, o MSX, o TK-2000, o Apple e o TRS-Color possuem um comando que permite detectar qualquer objeto na tela mais rapidamente do que pela checagem de suas coordenadas — mesmo que não conheçamos sua posição.

A COR PELO NÚMERO

Os comandos **ATTR** no Spectrum, **SCRN** no Apple e o **POINT** ou **PPOINT** no TRS-Color e no MSX devolvem, como resposta, a cor que está localizada em uma determinada posição (ou pixel, como é o caso do **PPOINT** no TRS-Color). Dessa maneira, para detectar a presença de qualquer objeto, basta atribuir-lhe uma cor e, depois, verificar todas as posições de tela.

No exemplo da bola, o círculo vermelho devolveria o número (código) relacionado à cor vermelha em todas as posições que aparecesse. Por meio da simples verificação desse código, poderíamos, por exemplo, fazer a bola bater no círculo e voltar.

O Spectrum devolve um número entre 0 e 255, levando em conta todos os **ATTR** (atributos) de cada posição de tela: as cores do **PAPER** e **INK**, se o **BRIGHT** está ligado ou não, e se o caractere se encontra sob a ação do comando **FLASH**. Os significados dos números nos atributos de cada posição da tela foram explicados na página 47.

O comando **SCRN** no Apple retorna um valor de 0 a 15, correspondente a uma cor da letra gráfica. Se obtivermos o número 13, por exemplo, para uma certa posição de tela, saberemos que naquela posição existe um caractere amarelo, pois 13 é o código estabelecido para a cor amarela.

O comando **POINT** no MSX devolve um número entre 0 e 16, correspondente à cor de um dos 256x192 pontos de sua tela. Este comando funciona igualmente bem tanto em alta quanto em baixa resolução gráfica.

O TRS-Color retorna um número entre -1 e 8, que se relaciona com a cor da posição de tela (ou pixel, para o caso do comando **PPOINT**). Obtemos o curioso resultado de -1 quando tentamos verificar a cor de uma letra: o texto só pode ser verde ou preto (e a cor preta é, justamente, o inverso da verde).

A sintaxe de cada um desses comandos é basicamente a mesma. Em todos eles, as coordenadas das posições de tela devem vir sempre entre parênteses. A única diferença é que, no **ATTR** do Spectrum, a coordenada Y aparece antes da X. Um exemplo seria:

```
PRINT ATTR(10,20)
```

onde 10 é a coordenada Y e 20 é a coordenada X da posição de tela cuja cor queremos saber.

Nos demais microcomputadores a ordem das coordenadas é a usual, ou seja, primeiro aparece a coordenada X e depois a Y. Para as mesmas coordenadas do exemplo anterior, teríamos:

```
PRINT SCRN(20,10)
```

para o Apple e

```
PRINT POINT(20,10)
```

para o MSX e o TRS-Color. O Spectrum possui um arquivo de atributos ao qual podemos fornecer, via comando **POKE**, diferentes valores — e, portanto, alterar o estado de certa posição de tela. Já o TRS-Color não tem um arquivo de cores separado, mas podemos mudar sua tela pela impressão (**PRINT**) de caracteres de outras cores.

Como exemplo do uso dos comandos **ATTR**, **SCRN** e **POINT**, digite e rode o seguinte programa:

S

```
 10 BORDER 0: PAPER 0: INK 9
 15 CLS
 20 FOR n=22528 TO 22559: POKE
 n,48: POKE n+672,48: NEXT n
 30 FOR n=22560 TO 23168 STEP
 32: POKE n,48: POKE n+31,48:
 NEXT n
 40 FOR n=1 TO 30: PRINT
 PAPER 6;AT INT (RND*8)*2+3,
 INT (RND*13)*2+3;" ": NEXT n
 50 LET x=15: LET y=10: LET xv
 =-1: LET yv=1
 80 PRINT AT y,x;"O": LET oy=y
 : LET ox=x
 90 LET x=x+xv: LET y=y+yv
140 IF ATTR (y,x-1)=48 OR ATTR
(y,x+1)=48 THEN  LET xv=-xv
145 IF ATTR (y-1,x)=48 OR ATTR
(y+1,x)=48 THEN  LET yv=-yv
190 PRINT AT oy,ox;" ": GOTO
 80
```

O display do Spectrum ocupa toda a tela: é um quadrado de bordas amarelas, contendo uma série de blocos também amarelos impressos aleatoriamente. Ao rodar o programa, o computador começa a movimentar a bola diagonalmente, para baixo e para a esquerda. Quando bate nas bordas ou em um dos blocos, ela volta, mas em outra direção.

COMO DETECTAR COLISÕES

Cabe ao comando **ATTR** verificar se a bola bateu nas bordas ou em algum dos blocos. Como estes são posicionados aleatoriamente na tela, só teríamos uma outra alternativa: guardar em variáveis as posições dos blocos em relação às coordenadas X e Y. Se usássemos duas variáveis para cada posição, ocuparíamos grande quantidade de memória. Além disso, precisaríamos incluir muitas verificações **IF...THEN** no programa, tornando-o muito lento.

Não enfrentaríamos os mesmos problemas se utilizássemos **IF...THEN** para verificar se a bola bateu nas bordas, já que suas coordenadas são facilmente calculadas. Mas, como recorremos ao comando **ATTR** para checar se a bola está batendo num bloco amarelo, é mais conveniente usá-lo também para as bordas, especialmente porque estas têm a mesma cor dos blocos.

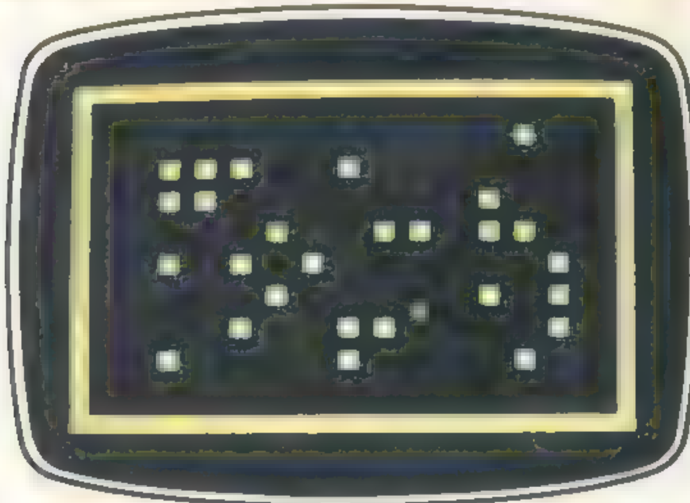
O programa começa definindo as cores da tela e criando a moldura amarela. As linhas 20 e 30, responsáveis pela moldura, inserem valores diretamente no arquivo de atributos, em vez de imprimirem espaços ou caracteres vazios. Dois laços **FOR...NEXT** percorrem os endereços de memória que contêm os atributos para posições ao redor da tela. Cada endereço recebe, via comando **POKE**, o número 48, que é o código para **PAPER** amarelo e **INK** preto (com **BRIGHT** e **FLASH** desligados).

A linha 40 imprime blocos aleatoriamente (desta vez, espaços). Os números randômicos que controlam a disposição dos blocos são projetados de maneira que pelo menos duas posições de tela os separem da borda (para se evitar que algum bloco seja desperdiçado, caindo na moldura).

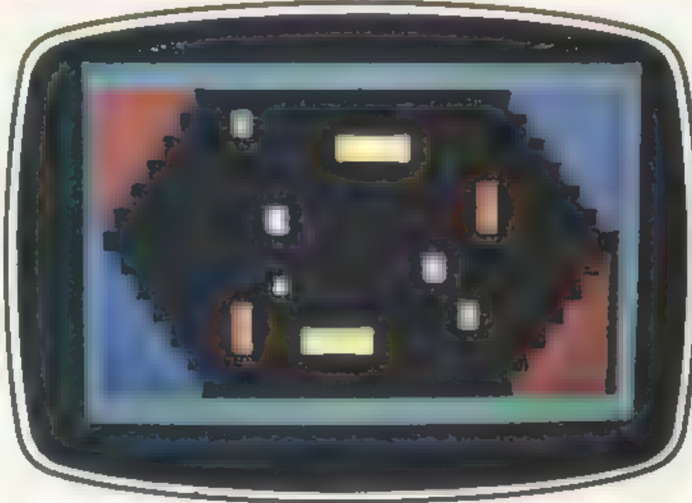
A linha 50 estabelece o valor inicial de algumas variáveis — **x** e **y** para a posição inicial da bola, e **xv** e **yv** para a direção inicial da bola nas coordenadas X e Y, respectivamente.

Em seguida, o computador salta para a linha 80, onde imprime a bola e define mais duas variáveis: **ox** e **oy**, que guardam os valores anteriores de **x** e **y**. Esses valores são usados para apagar a bola, uma vez que **x** e **y** já foram atualizados pelos cálculos que determinam o movimento da bola.

A linha 90 efetua os cálculos, somando os vetores de direção, determinados pelas variáveis **xv** e **yv**, às respectivas coordenadas **x** e **y**.



Display do Spectrum: os blocos são distribuídos aleatoriamente dentro de uma moldura quadrada.



Display do TRS-Color: apresenta cantos diagonais coloridos; a distribuiçao dos blocos não é aleatória.

O USO DO ATTR

As linhas 140 e 145 são as mais importantes, ja que executam a verificação da cor dos quatro quadrados que cercam a bola.

Como se vê no programa, a função **ATTR** assume a seguinte forma:

ATTR (coordenada y, coordenada x)

Mas este não é um comando direto — como **PRINT** e **LOAD**, por exemplo —, devendo sempre vir embutido em outro comando (no nosso caso, uma declaração **IF...THEN**).

Em ambas as linhas existem declarações **LET** logo após as condições. Elas simplesmente revertem o vetor de velocidade se o quadrado com o qual a bola bateu for amarelo (**ATTR** – 48 representa um quadrado amarelo).

Na prática, a bola bate no quadrado e volta num ângulo diferente de 90°. Ela deve estar vindo de cima ou de baixo (na direção y) ou de um dos lados do obstáculo (na direção x) no momento em que o atinge. Se ambos os vetores de velocidade fossem revertidos, a bola simplesmente voltaria pelo mesmo caminho. Por isso, a verificação só altera seu movimento na direção em que ela bate. Sua velocidade na outra direção (aquela em que ela não bateu em nada) continua sendo a mesma. Assim, se a bola está prestes a atingir um bloco amarelo logo abaixo, ela começa a se movimentar para cima, mas sua velocidade horizontal (na direção x) não é afetada.

Depois de realizar duas verificações, o Spectrum apaga a bola "velha", sobrepondo-a com um espaço vazio (linha 190), e volta para a linha 80 para continuar o movimento.



```
10 CLS0
20 PRINT STRING$(32,223);:PRINT
 @448,STRING$(32,223);
30 FOR K=1 TO 7:PRINT @32*K,CHR
$(223)+STRING$(7-K,191);
40 PRINT @32*K+24+K,STRING$(7-K
,175)+CHR$(223);
50 PRINT @448-32*K,CHR$(223)+ST
RING$(7-K,175);:PRINT @448-32*K
+24+K,STRING$(7-K,191)+CHR$(223
);
60 NEXT
70 FOR K=1 TO 16:READ A,B:POKE
1024+A,B:NEXT
80 DATA 137,143,114,159,111,159
,112,159,113,159,150,255,118,25
5,203,239,276,239,296,255,264,2
55,366,159,367,159,368,159,365,
159,406,143
90 X=32:Y=16:VX=2-RND(39)/10:VY
=2-RND(39)/10
100 SET(X,Y,5)
110 P1=POINT(X+VX,Y+VY):P2=POIN
T(X+VX,Y):P3=POINT(X,Y+VY):IF P
1=0 AND P2=0 AND P3=0 THEN 190
120 IF P2=0 AND P3=0 THEN 160
130 IF P2>0 THEN VX=-VX:IF P2<4
THEN VX=VX-SGN(VX)*RND(0) ELSE
IF P2>3THENVX=VX+SGN(VX)*RND(0)
140 IF P3>0 THEN VY=-VY:IF P3<4
THEN VY=VY-SGN(VY)*RND(0) ELSE
IF P2>3THENVY=VY+SGN(VY)*RND(0)
150 GOTO 190
160 VX=-VX:VY=-VY
170 IF PY>3 THEN VX=VX+SGN(VX)*
RND(0):VY=VY+SGN(VY)*RND(0)
180 IF P1<4 THEN VX=VX-SGN(VX)*
RND(0):VY=VY-SGN(VY)*RND(0)
190 IF ABS(VX)<1 THENVX=SGN(VX)
200 IF ABS(VX)>2 THEN VX=2*SGN(
VX)
210 IF ABS(VY)<1THEN VY=SGN(VY)
220 IF ABS(VY)>2 THEN VY=2*SGN(
VY)
230 RESET(X,Y):X=X+VX:Y=Y+VY
240 GOTO 100
```

O programa do TRS-Color exemplifica bem a utilidade do comando **POINT**. O display que aparece na tela desse computador é um pouco diferente daquele que aparece nos outros. Ele não contém blocos espalhados aleatoriamente, mas cantos diagonais coloridos. Usar variáveis para checar se a bola está batendo em algum desses cantos seria complicado e deixaria o programa muito lento. E, considerando que precisaríamos ainda adicionar vários **IF...THEN** para detectar colisões da bola com algum dos blocos no meio da tela, podemos concluir que tal processo é praticamente inviável.

VERIFICANDO MAIS DE UMA COR

O **POINT** permite a detecção de três quadrados ao redor da bola com poucas linhas de programa. E, se escolhermos cuidadosamente as cores, poderemos fazer a verificação de várias delas com apenas duas checagens.

O programa começa limpando a tela com um fundo preto e colocando os cantos, dois vermelhos e dois azuis (linhas 10 a 60). A linha 70 lê os dados dos blocos restantes (linha 80) e os imprime com **POKE**. Usando esse comando, em vez de **PRINT**, economizamos umas quatro linhas de programa.

A linha 90 determina as coordenadas iniciais da bola, **x** e **y**, e também as velocidades iniciais **xv** e **yv** — que, na verdade, são componentes do vetor de velocidade nas direções X e Y, respectivamente. A bola, representada por um ponto de baixa resolução, é então impressa nas posições x e y.

Para a checagem das cores, definem-se três variáveis (linha 110): **P1**, para o quadrado diagonalmente à frente da bola (qualquer que seja a direção em que ela esteja se movimentando), e **P2** e **P3**, para os quadrados nas posições x e y seguintes. Partindo do princípio segundo o qual não é necessário que sejam checados quadrados das direções em que a bola não se movimenta, concluímos que esses três testes já são suficientes.

O USO DO POINT

Como podemos ver na linha 110, o comando **POINT** toma a seguinte forma:

POINT (coordenada x, coordenada y)

Essa linha também é incumbida de verificar se os três quadrados são pretos (cor de número 0, que também é a cor de fundo). Em caso afirmativo, o computador vai para a linha 190, saltando os testes de reflexão, pois a bola não está batendo em nada.

Se as próximas posições x e y (direções horizontal e vertical, respectivamente) forem pretas, mas a diagonal seguinte não, a bola precisará voltar exatamente na mesma direção de que veio. Assim, o computador salta para a linha 160, que contém os procedimentos necessários para esse caso especial de colisão diagonal. Os dois vetores de velocidade (na direção x e na direção y) são invertidos, fazendo com que a bola retroceda.

As linhas seguintes também alteram a velocidade da bola, de acordo com a cor em que ela bate.

Se nenhum dos testes executados até a linha 130 for verdadeiro, a bola deve estar atingindo algum objeto (seja ele um canto, seja ele um bloco) pelas laterais. A linha 130 verifica se o quadrado atingido está na horizontal (lados direito ou esquerdo) e, constatando que sim, inverte o vetor relevante da velocidade. O vetor relevante é oposto ao eixo de colisão — portanto, no nosso exemplo, o vetor y é invertido. Em seguida, o computador efetua o teste a fim de identificar a cor do quadrado atingido.

ACELERAÇÃO DA BOLA

Os números 1, 2 e 3 correspondem ao verde, ao amarelo e ao azul. Se a cor do quadrado atingido for uma destas, a velocidade será diminuída; se for vermelho, cinza, ciano, magenta ou laranja (número maior que 3), a bola será, então, acelerada.

A vantagem de se escolher as cores (ou categoria de cores) com cuidado é evidente. Elas se agrupam em duas categorias: uma que causa diminuição de velocidade (1, 2 e 3) e outra que causa aumento (4 a 8). Ao selecionar números próximos em cada categoria, diminuímos a quantidade de comandos **IF...THEN**, economizamos memória e agilizamos o programa.

A linha 140 faz basicamente a mesma coisa que a 130, só que efetua a checagem na vertical. Depois, o computador salta para a linha 190, onde se inicia a rotina que verifica o valor da velocidade. Se esta for maior que 2, obtém-se um estranho efeito na tela, pois, para simular movimento, o computador salta sobre um quadrado. Um salto muito grande pode, eventualmente, fazer a bola ultrapassar um bloco colorido sem o perceber. Daí a importância da rotina que checa a velocidade.

O mesmo tipo de problema aconteceria com uma velocidade menor que 1; mas um outro teste cuida disto. Observe que todos os valores estão em módulo (**ABS**), ou seja, seus sinais são ignorados. Embora -1 seja menor que 1, a magnitude da velocidade é a mesma para ambos os valores — o sinal menos indica que o sentido é contrário. O **ABS** simplesmente elimina o sinal antes de testar o valor da velocidade.

A linha seguinte, isto é, a 230, atribui à bola a cor de fundo, apagando, assim, sua última posição. Depois, o programa atualiza a velocidade da bola e manda o computador de volta para a linha 100. Esta dá continuidade ao movimento da bola.



```
10 SCREEN 3:COLOR 15,1,3·R=RND(
-TIME)
20 LINE (0,96)-(96,0),6:PAINT (
0,0),6
30 LINE (255,96)-(160,191),6:PA
INT (255,191),6
40 LINE (0,96)-(95,191).4:PAINT
 (0,191),4
50 LINE (255,96)-(158,0),4:PAIN
T (255,0),4
60 LINE (0,0)-(255,0),3:LINE-(2
55,191).3:LINE-(0,191).3:LINE-(
0,0),3
70 FOR I=1 TO 8:READ X1,Y1,X2,Y
2,C:LINE (X1,Y1)-(X2,Y2),C,BF:N
EXT
80 DATA 130,30,160,45,11,100,17
0,130,155,11,80,30,95,65,2,170,
```



```
135,180,160,2,90,90,100,120,13,
150,80,160,110,13,60,100,70,140
,9,190,55,200,95,9
90 X=120+16*RND(1):Y=90+16*RND(
1):VX=4*(-1)^INT(10*RND(1)):VY=
4*(-1)^INT(10*RND(1))
100 PSET(X,Y)
110 P1=POINT(X+VX,Y+VY):P2=POIN
T(X+VX,Y):P3=POINT(X,Y+VY):IF P
1=1 AND P2=1 AND P3=1 THEN 190
115 PLAY "O5A64"
120 IF P2=1 AND P3=1 THEN 160
130 IF P2>1 THEN VX=-VX
140 IF P3>1 THEN VY=-VY
150 GOTO 190
160 VX=-VX:VY=-VY
190 PRESET(X,Y):X=X+VX:Y=Y+VY
240 GOTO 100
```

A primeira linha seleciona a tela de baixa resolução, define as cores e ativa o gerador de números randômicos.

A tela do MSX não contém blocos espalhados aleatoriamente, mas cantos diagonais coloridos. Utilizar variáveis para checar se a bola está batendo em algum desses cantos seria complicado e deixaria o programa extremamente lento. E, considerando que precisaríamos ainda adicionar vários **IF...THEN** para detectar colisões da bola com algum dos blocos no meio da tela, podemos concluir que tal processo é praticamente inviável.

VERIFICANDO MAIS DE UMA COR

Com algumas linhas de programa, o **POINT** possibilita a detecção de três quadrados ao redor da bola. E, se escolhermos cuidadosamente as cores, poderemos fazer a verificação de várias delas com apenas duas checagens.

O programa começa ajustando o fundo: limpa a tela com um fundo preto, coloca os cantos — dois vermelhos e dois azuis — e desenha uma moldura verde (linhas 10 a 60). A linha 70 lê os dados dos blocos restantes que estão na linha **DATA** 80 e os coloca na tela via **LINE**. Como você pode observar, este programa é um bom exemplo de como os comandos gráficos funcionam em alta e em baixa resolução.

A linha 90 determina as coordenadas iniciais da bola, **X** e **Y**, e as velocidades iniciais, **VX** e **VY**, que são componentes do vetor de velocidade nas direções X e Y, respectivamente. A bola, representada por um ponto de baixa resolução, é, então, impressa nas coordenadas X,Y.

Para a checagem das cores, definem-se três variáveis (linha 110): **P1**, para o quadrado diagonalmente à frente da bola (qualquer que seja a direção em que ela esteja se movimentando), **P2** e **P3**, para os quadrados nas posições de baixa resolução seguintes (cada bloco de baixa resolução tem quatro por quatro pontos). Partindo do princípio de que não é preciso checar os quadrados das direções nas quais a bola não se movimenta, concluímos que esses três testes são suficientes.

A linha 110 também verifica se os três quadrados são pretos (cor de número 1, que também é a cor de fundo). Em caso afirmativo, o computador vai para a linha 190, saltando os testes de reflexão, pois a bola não está batendo em nada.

Se a posição x de baixa resolução seguinte (quatro pontos adiante na direção horizontal) e a posição y seguinte (quatro pontos na direção vertical) forem pretas, mas a próxima não, a bola precisará voltar exatamente na mesma direção de que veio. Assim, o computador salta para a linha 160, que contém os procedimentos necessários para esse caso especial de colisão diagonal. Os dois vetores de velocidade (na direção x e na direção y) são invertidos, fazendo a bola retroceder.

Se nenhum teste executado até a linha 130 for verdadeiro, a bola deve estar atingindo algum objeto (seja ele um canto ou um bloco) pelas laterais. A linha 130 verifica se o quadrado atingido está na horizontal (lados direito ou esquerdo), e, constatando que sim, inverte o vetor relevante da velocidade. O vetor relevante é oposto ao eixo de colisão — portanto, no nosso exemplo, o vetor y é invertido.

A linha seguinte, ou seja, a 190, atribui à bola a cor de fundo, apagando, assim, sua última posição. Depois, o programa atualiza a velocidade da bola e manda o computador de volta para a linha 100. Esta dá continuidade ao movimento da bola.

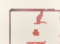


```
10   HOME : GR : COLOR= 13
20   FOR X = 0 TO 39: PLOT X,0:
```

```
PLOT X,39: NEXT
30  FOR Y = 0 TO 39: PLOT 0,Y:
PLOT 39,Y: NEXT
40  FOR N = 1 TO 30
50 RX =  INT ( RND (1) * 34) +
3
60 RY =  INT ( RND (1) * 34) +
3
70  PLOT RX,RY: NEXT
80 X = 19:Y = 19:XV =  - 1:YV =
 1
90  COLOR= 3: PLOT X,Y:OX = X:O
Y = Y
100 X = X + XV:Y = Y + YV
110 TX =  SCRN( X + XV,Y)
120 TY =  SCRN( X,Y + YV)
130  IF TX = 13 THEN XV =  - XV

140  IF TY = 13 THEN YV =  - YV

150  COLOR= 0: PLOT OX,OY
160  GOTO 90
```



Para o TK-2000, substitua os números 13, no programa do Apple, por 5. As linhas alteradas ficam assim:

```
10  HOME : GR : COLOR= 5
130  IF TX = 5 THEN XV =  - XV
140  IF TY = 5 THEN YV =  - YV
```

O programa para o Apple e o TK-2000 não difere muito daquele do Spectrum. Ele começa limpando a tela, definindo o modo gráfico de baixa resolução e especificando a cor amarela (número de código 13), tanto para os blocos como para as bordas. Em gráficos de baixa resolução temos uma tela inicial de 40x40, com uma janela para texto. Se indicarmos algum ponto da tela, aparecerá naquela posição um retângulo na cor que estiver em vigor. No TK-2000, o código da cor é 5 (vermelho).

A linha 20 é responsável pelas partes superior e inferior das bordas. Observe que mantivemos o valor da coordenada y constante e variamos o valor da coordenada x, traçando, assim, as duas linhas horizontais.

A linha 30 segue o mesmo princípio da linha 20, mas varia os valores da coordenada y e traça os lados esquerdo e direito das bordas.

Os blocos espalhados aleatoriamente pela tela são gerados entre as linhas 40 e 70. A linha 50 atribui à coordenada **RX** um valor randômico entre 3 e 36, e a linha 60 atribui a **RY** um valor nesse mesmo intervalo.

RX e **RY** são as coordenadas do bloco a ser plotado (traçado no gráfico). A linha 80 define as coordenadas iniciais do retângulo que, no caso, representa a bola, e também ajusta os sentidos dos vetores de velocidade **XV** e **YV**.

A linha 90 define a cor da bola — o número 3 é o código para a cor púrpura no Apple e branca no TK-2000 —, imprimindo-a, em seguida, nas coordenadas x e y. A mesma linha define duas outras variáveis importantes: **OX** e **OY**, que armazenam as coordenadas da posição anterior da bola para que esta possa ser apagada mais tarde.

A posição da bola é atualizada pela linha 100 do programa, que incrementa as coordenadas da bola na direção do vetor da velocidade.

A cor do quadrado situado imediatamente ao lado da bola — ou seja, aquele no qual a bola está prestes a bater pela horizontal — é guardada em **TX**, por meio do comando **SCRN**. A cor do quadrado logo acima ou abaixo (depende da trajetória) da bola é guardada na variável **TY**. As linhas 130 e 140 verificam se esses quadrados que estão no caminho da bola são amarelos (cor 13) ou, no caso do TK-2000, vermelhos (cor 5). Se o quadrado da horizontal for amarelo ou vermelho, inverte-se o vetor de velocidade horizontal da bola. Tratando-se do quadrado de cima ou de baixo, o vetor invertido será o da componente vertical.

Para apagar a bola, utilizamos o já conhecido artifício de redesenhá-la na mesma posição, só que na cor de fundo. Para isso, a linha 150 seleciona a cor 0 (preta), que é a cor de fundo, e plota um quadrado no local definido pelas coordenadas **OX,OY** — coordenadas "velhas" da bola.

O programa continua na linha 160, que envia o computador de volta para a linha 90 e tudo se repete.

Observe que o programa não inclui testes para a verificação de quadrados dispostos diagonalmente à trajetória percorrida pela bola. Assim, eles são ignorados e eliminados. Não é difícil acrescentar uma linha que faça esse teste, mas o efeito seria uma bola indo e voltando pelo mesmo caminho.



É muito fácil transformar os programas em jogos, bastando adicionar-lhes algumas linhas. O programa de TRS-Color, por exemplo, constituiria uma boa base para um jogo tipo fliperama. Podemos fazer o computador executar várias operações depois das condições **IF...THEN** — por exemplo, soar um bip cada vez que a bola atinge um bloco ou borda, ou mesmo acrescentar um placar. Outros artigos de INPUT utilizarão os comandos vistos aqui em diversas rotinas de jogos.



O TRS-Color não dispõe apenas do **POINT**, usado na tela de baixa resolução. Há outro comando, o **PPOINT**, que pode ser utilizado na tela de alta resolução. No MSX, o mesmo comando **POINT** se aplica à alta resolução.

Em vez de devolver o código da cor de um certo quadrado, como faz o comando **POINT**, o **PPOINT** devolve o código da cor de um ponto.

Digite e rode o programa a seguir e veja como empregar esse comando.



```
10 PMODE 1,1
20 PCLS 3
30 SCREEN 1,0
40 LINE(20,20)-(235,171),PRESET
,BF
50 VX=RND(7)-4:VY=RND(7)-4
60 BX=127:BY=95
70 PSET(BX,BY,4)
80 IF PPOINT(VX+BX,BY)=3 THEN V
X=RND(3)*((VX>0)-(VX<0))
90 IF PPOINT(BX,BY+VY)=3 THEN V
Y=RND(3)*((VY>0)-(VY<0))
100 PRESET (BX,BY)
110 BX=BX+VX:BY=BY+VY
120 GOTO 70
```



```
10 SCREEN 2:COLOR 1,15,15:R=RND
(-TIME)
40 LINE (16,16)-(238,177),4,B:P
AINT(0,0),4
50 VX=INT(RND(1)*8-4):VY=INT(RN
D(1)*8-4)
60 BX=127:BY=95
70 PSET(BX,BY),4
80 IF POINT(VX+BX,BY)=4 THEN VX
=-VX
90 IF POINT(BX,BY+VY)=4 THEN VY
=-VY
100 PRESET(BX,BY),15
110 BX=BX+VX:BY=BY+VY
120 GOTO 70
```

Uma bola movimenta-se pela tela, batendo e voltando. O comando **PPOINT** — ou **POINT**, no MSX — é usado para verificar se a bola bateu na parte azul. As linhas 80 e 90 checam se o ponto à frente da bola é azul. Em caso afirmativo, a bola é revertida pelo restante das duas linhas, que calculam uma velocidade aleatória para o movimento da bola (no programa do MSX a velocidade é apenas invertida).

Em geral, usa-se **PPOINT** (ou **POINT**, no MSX) para detectar colisões. Se tivermos, por exemplo, um gráfico colidindo com outro, podemos checar a colisão verificando a cor de um deles.

MÚSICA EM SEU MICRO



- NOTAS E ESCALAS
- TONS E SEMITONS
- PEQUENAS MELODIAS
- TRANSFORME O COMPUTADOR EM UM ÓRGÃO MUSICAL

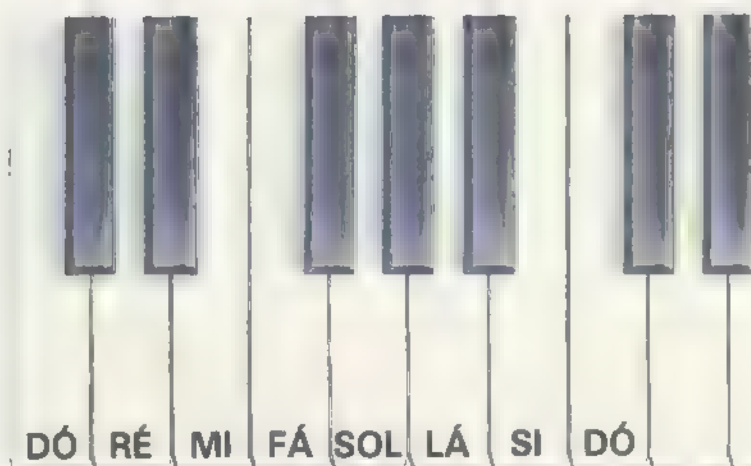
Bach ou Beethoven? Pink Floyd ou The Cure? Seja qual for a sua escolha musical, você pode tornar-se um [illegible] instrumentista, dedilhando [illegible] teclado de um microcomputador.

Com exceção do Apple, todos os microcomputadores mencionados neste artigo contam com algum tipo de comando capaz de produzir sons. O funcionamento de comandos sonoros, aliás, já foi explicado anteriormente em *Quebre a Barreira do Som* (página 168). Um artigo posterior — *Apple e TK-2000: Efeitos Sonoros* — abordou o funcionamento de uma rotina em código de máquina, que será utilizada no programa deste artigo; ela permite que os micros dessas linhas produzam sons.

De um modo geral, é possível selecionar tanto a tonalidade como a duração da nota emitida. No MSX e no TRS-Color, outros parâmetros também podem ser modificados.

A forma de produzir sons por meio de um programa em BASIC varia de computador para computador. No MSX, podemos tocar até três notas ao mesmo tempo, o que possibilita a execução de acordes musicais. Os demais produzem só uma nota por vez. O MSX permite que controlemos outros parâmetros essenciais para certas aplicações.









Nenhuma dessas diferenças é relevante no momento. De fato, melodias simples podem ser executadas em qualquer um dos cinco computadores abordados neste artigo, que pretende ser apenas um guia para principiantes. Começaremos com algumas noções de teoria musical, necessárias à compreensão do programa, que transforma parte do teclado de seu micro em um instrumento simples. Em artigo posterior, veremos de que maneira tirar maior proveito dos recursos musicais do computador.

Antes de iniciar, devemos explicar o significado de dois termos fundamentais. O primeiro deles é a expressão "altura", que aqui quer dizer o mesmo que tom ou tonalidade. Ele permite classificar as notas musicais em graves e agudas. Assim, uma nota mais alta que outra será mais aguda que ela; uma nota mais baixa será mais grave. O segundo termo, "intervalo", indica a distância musical entre duas notas de tonalidade diferente.

UMA ESCALA SIMPLES

Uma escala é uma série ascendente de notas musicais, cada uma mais alta que a anterior. Embora em computação seja tecnicamente melhor recorrer a letras do alfabeto — de A a G — para designar as notas (como acontece na Europa e nos Estados Unidos), o sistema adotado no Brasil emprega os nomes tradicionais — dó, ré, mi, fá, sol, lá, si. Assim, uma escala começa com dó e vai "subindo" até o próximo dó, num total de oito notas.

Essa escala "dó-ré-mi" é chamada de "escala maior". Ela é definida por uma relação musical entre as notas. Assim, qualquer seqüência de notas com o mesmo padrão de intervalos musicais entre si é uma "escala maior".

E AS TECLAS PRETAS?

As teclas brancas de um piano correspondem às notas (letras) A, B, C, D, E, F e G, num ciclo que se repete pelo teclado afora — C é dó, D é ré, F é mi etc. A "escala maior" corresponde a oito teclas, começando num C e terminando no próximo C. Essa escala é denominada escala de "C maior", por motivos óbvios. Como dissemos antes, uma "escala maior" pode iniciar-se em qualquer tonalidade, mas a escala de "C maior" deve começar necessariamente num C, que tem uma altura específica. A escolha dessa letra como primeira nota fixa a tonalidade da escala.

Para que servem então as teclas pretas do teclado? Acontece que a "escala maior" não contém todas as notas possíveis entre a primeira e a última letras. Entre dó e ré, por exemplo, existe uma nota que não pertence à "escala maior"; o mesmo ocorre com outros pares de notas. As teclas pretas correspondem a esse tipo de som intermediário. Assim, a tecla preta entre C e D (dó e ré em "C maior") é chamada de C sustenido ou D bemol ("sustenido" significa mais alto e "bemol", mais baixo), conforme usemos a nota acima ou abaixo dela como referência.

A nota entre D e E é D sustenido ou E bemol, e assim por diante. Note que não há tecla preta entre E e F (mi e fá em "C maior") nem entre B e C (si e dó em "C maior").

O intervalo existente entre duas teclas vizinhas (mesmo que elas tenham cores diferentes) é chamado de semitom, de maneira que os intervalos entre C e C sustenido, C sustenido e D e entre E e F são todos semitons. Um intervalo que contenha dois semitons é denominado tom "inteiro".

O arranjo de tons e semitons que definem uma escala maior é:

dois semitons = um tom entre dó e ré (C e D em "C maior").

dois semitons = um tom entre ré e mi (D e E em "C maior")

um semitom entre mi e fá (E e F em "C maior").

dois semitons = um tom entre fá e sol (F e G em "C maior").

dois semitons = um tom entre sol e lá (G e A em "C maior").

dois semitons = um tom entre lá e si (A e B em "C maior").

um semitom entre si e dó (B e C em "C maior").

OUTRAS ESCALAS MAIORES

Segundo o que foi exposto, existem doze semitons entre o primeiro e o segundo dó de uma "escala maior" (total de oito notas).

Suponhamos agora que precisamos de uma "escala maior" em outra tonalidade — começando em G, por exemplo. Nela, G seria dó e A seria ré; mas, se tocássemos as seis teclas seguintes, o arranjo de intervalos não seria o de uma "escala maior": os intervalos entre lá e si, e si e dó não estariam corretos. Para obtermos uma "escala maior" verdadeira, F deve ser substituído por F sustenido (ou si, em nossa nova escala), restabelecendo o padrão correto de intervalos. Para entender melhor, estude o diagrama da página anterior ou use um teclado de verdade.

Uma "escala maior" começando em F seria então: F, G, A, B bemol, C, D, E e F. (B deve ser bemol, caso contrário a nota fá e os intervalos mi-fá e fá-sol estarão errados).

Podemos obter uma "escala maior" começando em qualquer nota, desde que respeitemos o arranjo de intervalos. Mas qualquer escala não iniciada em C deverá usar uma ou mais teclas pretas: apenas a escala "C maior" usa só teclas brancas. Fica explicado porque ela é tão popular: é mais fácil tocar usando só as teclas brancas.

Muitas músicas simples — como cantigas de ninar, músicas folclóricas e alguns hinos — podem ser tocadas na escala maior. A canção *Três Ratinhos Cegos*, por exemplo, começa assim:

mi, ré, dó
mi, ré, dó
sol, fá, fá, mi
sol, fá, fá, mi...

e as primeiras notas de *Atirei o Pau no Gato* são:

sol, lá, mi, ré, mi, fá, sol, sol,
sol, fá, sol, lá, lá, lá,
sol, lá, mi, mi, mi...

Escrevemos aqui apenas as tonalidades; o ritmo fica por conta do leitor. Só com notação musical — um tema que não abordaremos agora — poderíamos transmitir a informação completa a respeito da melodia — altura, ritmo, intensidade, entre outras coisas.

Músicas mais complexas não poderão ser executadas somente com as teclas brancas. Geralmente, começamos em uma escala e, com o desenvolvimento da música, passamos temporariamente para outras escalas, usando notas que não aparecem na primeira.

Veja, por exemplo, o caso de **Yesterday**, música composta por John Lennon e Paul McCartney que se tornou um dos grandes sucessos dos Beatles:

ré, dó, dó, mi, fá, fá sustenido, sol sustenido, lá, si, dó, si, lá, lá...

Essas notas, estranhas à escala original, são um requinte; elas dão um toque especial a muitas melodias.

BEMÓIS OU SUSTENIDOS?

Por que se usa o termo "sustenido" (em F sustenido) na escala "G maior", e "bemol" (em B bemol) na escala "F maior"? Afinal, F sustenido e G bemol correspondem à mesma tecla preta. Por que não usar só um dos termos, deixando o outro de lado? Quem tentar criar novas "escalas maiores" — começando em notas diferentes de C — terá a resposta. Na verdade, os dois termos são necessários para garantir que, em qualquer escala, cada nota seja representada por uma letra diferente. Assim, a substituição do B bemol da escala "F maior" por A sustenido, resultaria em: F, G, A, A sustenido, C, D, E e F. Em tal seqüência de notas aparecem dois A e nenhum B, o que pode gerar confusão.

Da mesma maneira, a substituição de F sustenido por G bemol em "G maior", resultaria em uma escala com dois G e nenhum F. Já o emprego de bemóis no lugar de sustenidos, ou vice-versa, é tão deselegante como erros de ortografia em um texto. Programas para compor música que usem só bemóis — ou sustenidos —, com a justificativa de que C sustenido é igual a D bemol, deixam muito a desejar.

FREQÜÊNCIA E INTERVALOS

Som é o efeito produzido pelas vibrações no ar. Quanto mais rapidamente estas se repetirem, ou quanto maior for sua freqüência, mais aguda será a nota correspondente. A unidade usada para exprimir freqüência é cps (ciclos por segundo) ou Hz (hertz, que significa a mesma coisa).

Se tomarmos a nota C, cuja frequência é igual a 256 Hz, e dobrarmos esse valor, obteremos outra nota C uma oitava acima da primeira — mais aguda, portanto (uma oitava é o intervalo entre o dó mais baixo e o dó mais alto). Dobrando novamente o valor da freqüência, ele passará de 512 para 1024 Hz, que corresponde à próxima nota C — uma oitava acima da anterior. Desse modo, quando *multiplicamos* a freqüência de uma nota, *adicionamos* a ela um intervalo musical.

Já vimos que há doze semitons em uma oitava (treze teclas, oito brancas e cinco pretas, com doze semitons *entre* si). Se, dobrando a freqüência, subimos uma oitava, por que fator devemos multiplicar a freqüência de uma nota para subir um semitom? No caso de uma oitava acima, temos a equação:

freqüência da nota x 2 = frequência da nota uma oitava mais alta.

Chamamos de X o fator necessário para subir um semitom. Teremos então:

frequência da nota
xXxXxXxXxXxXxXxXxXxXxXxX =
freqüência da nota uma oitava mais alta.
*X*X*X*X*X*X*X*X*X*X*X*X =

O número que divide a razão 2:1 em doze partes iguais é a décima segunda raiz de 2 (para calcular esse valor em BASIC, use a expressão 2↑(1/12)). Se multiplicarmos 256 por esse valor, obteremos a nota um semitom mais alta. Uma nova multiplicação adiciona mais um semitom. Se repetirmos o processo, após doze multiplicações consecutivas, chegaremos à freqüência da oitava seguinte. Por tudo isso, a décima segunda raiz de dois é uma constante fundamental da música.

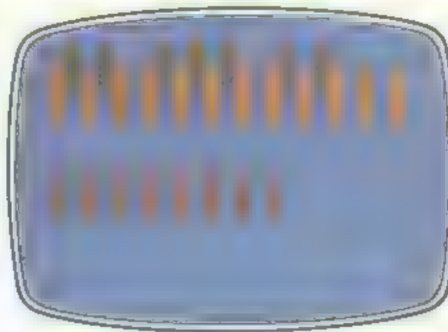
MELODIAS NA "ESCALA MAIOR"

Se quisermos tocar uma música usando o programa do final deste artigo, teremos que adivinhar qual nota é dó, qual é ré, e assim por diante. A questão é: como tocar "de ouvido"?

Existirá algum método para fazer com que a música possa ser tocada somente nas teclas brancas, facilitando sua execução? Neste caso também, um caminho possível é descobrir qual nota da música é um dó, deduzindo, a partir daí, as posições das demais notas. Muitas melodias começam ou terminam com essa nota; além disso, o dó acaba funcionando como o "centro de gravidade" de certas músicas simples. Uma vez encontrada, essa nota nos servirá de guia, e, com um pouco de sorte, descobriremos as outras. Tente tocar a música desde o início, nota por nota; preste muita atenção no que estiver ouvindo, procurando perceber se a tonalidade está subindo ou descendo naquele ponto e se a nota seguinte se encontra na tecla vizinha, ou mais além.

Se nos enganarmos sobre qual é a nota dó, algumas notas parecerão muito graves ou muito agudas e teremos de procurar o dó "certo". Com paciência e perseverança, porém, qualquer um pode desenvolver uma espécie de "intuição musical" e alguns serão capazes de tocar músicas inteiras no teclado de um microcomputador.

Para facilitar o trabalho dos principiantes, contudo, este artigo menciona as notas de algumas melodias mais ou menos conhecidas.



É assim que o diagrama do teclado aparece na tela dos micros da linha TRS-Color.

UM TECLADO MUSICAL

Os programas a seguir transformam a parte superior do teclado do seu computador em um pequeno órgão. As letras Q, W, E, R, T, Y, U, I serão as notas dó, ré, mi etc. As teclas da direita avançam na oitava seguinte. As teclas 2, 3, 5, 6, 7 funcionam como as teclas pretas do piano. Um diagrama de seu teclado fica mais ou menos assim:

2 3 5 6 7 etc.
Q W E R T Y U I etc.

d r m f s l s d
o é i a o á i ó

O programa coloca na tela um diagrama semelhante para orientar o tecladista. Além disso, quando forem tocadas as notas da primeira oitava da "escala maior", seus nomes aparecerão num canto do vídeo.

Digite a seguir o programa específico para o seu micro e siga as instruções que o acompanham para se tornar um "virtuose".



O programa do Spectrum transforma as duas linhas superiores do teclado em um órgão. Para ouvir uma nota, basta pressionar a tecla correspondente.

A listagem começa estabelecendo os valores das variáveis que correspondem à duração das notas, ao endereço do laço principal (guardado na variável **loop**, para economizar memória), e à coordenada x do comando **PRINT AT**, usado para impressão das notas na tela. Ela continua selecionando as cores da tela, e entra a seguir na porção responsável pela produção de sons.

A variável **loop** contém o número da linha para o qual o programa retornará após executar uma nota; ele aguardará então que você pressione a tecla seguinte. Quando isso acontecer, o código do caractere correspondente será colocado na variável **nota**, com o auxílio do comando **CODE**.

O computador irá então para a linha cujo número está em **nota**. Os números de linha correspondem aos códigos das teclas usadas (de 48 a 121). O comando **SOUND** em cada uma dessas linhas é ajustado para tocar a nota correspondente à tecla (Q, por exemplo, corresponde a dó). Se consultarmos o manual — ou o artigo *Quebre a Barreira do Som* —, veremos que o primeiro número após **SOUND** corresponde à duração da nota (aqui representada pela variável **d**) e o segundo, à tonalidade.

Quando a tecla apertada for de uma nota da "escala maior", o nome desta também será impresso na tela, pela mesma linha que a executa.

Depois de emitir o som, o computador retorna à linha 1000, onde espera uma nova tecla ser pressionada.

```
  1 GOTO 900
 47 GOTO loop
 48 SOUND d,15: GOTO loop
 50 SOUND d,1: GOTO loop
 51 SOUND d,3: GOTO loop
 53 SOUND d,6: GOTO loop
 54 SOUND d,8: GOTO loop
 55 SOUND d,10: GOTO loop
 57 SOUND d,13: GOTO loop
101 PRINT AT y,x;"MI": SOUND d
,4: GOTO loop
105 PRINT AT y,x;"DO": SOUND d
,12: GOTO loop
111 SOUND d,14: GOTO loop
112 SOUND d,16: GOTO loop
113 PRINT AT y,x;"DO": SOUND d
,0: GOTO loop
114 PRINT AT y,x;"FA": SOUND d
,5: GOTO loop
116 PRINT AT y,x;"SO": SOUND d
,7: GOTO loop
117 PRINT AT y,x;"SI": SOUND d
,11: GOTO loop
119 PRINT AT y,x;"RE": SOUND d
,2: GOTO loop
121 PRINT AT y,x;"LA": SOUND d
,9: GOTO loop
800 GOTO loop
900 LET D=.03: LET X=5: LET lo
op=1000
901 BORDER 4: PAPER 4: CLS
902 PRINT AT 8,6;"d r m f s l
s d"
903 PRINT AT 9,6;"o e i a o a
i o"
910 LET a$=" 2 3   5 6 7   9 0
"
920 FOR y=3 TO 4: GOSUB 990
930 NEXT y
940 LET a$="Q W E R T Y U I O
P"
941 PAPER 7: INK 0
950 FOR y=4 TO 6: GOSUB 990
```

```
 960 NEXT y
 980 LET x=15: LET y=15: GOTO 1
000
 990 FOR i=1 TO LEN a$
 991 IF a$(i)<>CHR$ 32 THEN
PRINT AT y,x+i;a$(i);
 992 NEXT i: RETURN
1000 PRINT AT y,x;CHR$ 32;CHR$
32
1005 LET a$=INKEY$: IF a$="" TH
EN  GOTO 1000
1100 LET nota=CODE a$: GOTO not
a
```

Aqueles que possuem micros da linha Spectrum importados devem substituir o comando **SOUND** pelo comando **BEEP**, que tem a mesma sintaxe.

Após iniciar a execução do programa, mantenha uma das teclas pressionadas para ouvir um som pulsante correspondente à auto-repetição da tecla. O som não é emitido continuamente, como num órgão, pois o comando **SOUND** produz sons com duração limitada. Assim, o que o programa realmente faz é tocar a mesma nota repetidas vezes.

Vejamos agora o que acontece com os modelos da linha MSX.

Os microcomputadores dessa linha contam com poderosos comandos sonoros graças a um microprocessador interno dedicado inteiramente à produção de sons, e capaz de funcionar simultaneamente com a unidade central. Para programá-lo em BASIC, precisamos recorrer a vários comandos **SOUND**, com o propósito de definir diversos parâmetros do som — ou ruído — que estamos querendo emitir. Uma introdução a esse assunto foi apresentada no artigo *Quebre a Barreira do Som*, onde abordamos vários sons não-musicais.

Contudo, para transformar a parte superior do teclado num programa semelhante ao dos outros micros, podemos usar o comando **PLAY**, que torna o programa mais simples e curto.

A primeira linha do programa estabelece os parâmetros iniciais das notas: volume, duração, velocidade de execução. As linhas 20 e 30 colocam todas as notas usadas pelo programa dentro da variável indexada **N$**, recorrendo ao comando **READ** e aos dados da linha **DATA**. A linha 40 cria um cordão contendo todas as teclas usadas e um outro que permite dar nome a algumas das notas (dó, ré, mi etc). As linhas que vão de 50 a 100 cuidam da organização do vídeo. A primeira delas seleciona a tela de texto com 32 colunas, escolhe as cores e desliga as teclas de função.

Nas três linhas seguintes — ou seja, 55, 60 e 65 —, o comando **VPOKE** é utilizado para modificar a tabela de cores — **BASE (6)** —, permitindo, assim, a impressão de letras coloridas. Se você não gostar das cores, pode mudar os códigos nos locais onde eles aparecem — para maiores informações, veja o artigo *Os Comandos PEEK e POKE* (página 261).

Na linha 70 , o comando **VPOKE** é novamente utilizado, desta vez para escrever na tela. O **FOR...NEXT** contido nessa linha imprime parte do diagrama do teclado na tela, obtendo com **READ** os códigos dos caracteres de cada uma delas na linha 75 e modificando a tabela de nomes — **BASE(5)**. O restante do diagrama é feito do mesmo modo pelas linhas 80 e 85.

As linhas 90 e 100 completam o quadro, imprimindo o nome das notas sob as teclas correspondentes.

A linha 120 espera que acionemos uma tecla. A 130 descobre qual das notas do cordão **M$** corresponde à tecla pressionada; se a tecla não pertencer ao conjunto predefinido, a linha 120 será repetida.

A linha 140 é a que realmente executa a nota, usando o **PLAY**. A altura da nota é selecionada pela variável **NT**, que é usada para escolher o elemento de **N$** que contém os operandos adequados do **PLAY**.

A seguir, um comando **ON GOTO** na linha 150 imprimirá o nome da nota no canto superior esquerdo da tela, se ela pertencer à primeira oitava de "C maior". O programa retorna então para verificar uma nova tecla.

```
10 PLAY "V15L64T255"
20 DIM N$(21):FOR K=0 TO 16:REA
D N$(K):NEXT
30 DATA C,C#,D,D#,E,F,F#,G,G#,A
,A#,B,O4C,O4C#,O4D,O4D#,O4E
40 M$="Q2W3ER5T6Y7UI9O0P":D$="Q
WERTYUIO"
50 SCREEN 1:COLOR 1,6,6:KEY OFF
55 FOR I=6 TO 7:VPOKE BASE(6)+I
,15*16+1:NEXT
60 FOR I=8 TO 11:VPOKE BASE(6)+
I,16+15:NEXT
65 FOR I=12 TO 15:VPOKE BASE(6)
+I,16+10:NEXT
70 FOR K=0 TO 6:READ A:N=2*A-20
*INT(A/10):VPOKE BASE(5)+71+N,A
:VPOKE BASE(5)+103+N,A:NEXT
75 DATA 50,51,53,54,55,57,48
80 FOR K=0 TO 9:READ A:VPOKE BA
SE(5)+102+K*2,A:VPOKE BASE(5)+1
34+K*2,A:VPOKE BASE(5)+166+K*2,
A:NEXT
85 DATA 81,87,69,82,84,89,85,73
,79,80
90 LOCATE 4,9:PRINT "d r m f s
l s d"
100 LOCATE 4,10:PRINT "o e i a
o a i o"
120 A$=INKEY$:IF A$="" THEN 120
130 NT=INSTR(M$,A$):IF NT=0 THE
N 120
140 PLAY "O3"+N$(NT-1)
150 LOCATE 0,0:D=INSTR(D$,A$):O
N D GOTO 170,180,190,200,210,22
0,230,170
160 PRINT "  ":GOTO 120
170 PRINT "DO":GOTO 120
180 PRINT "RE":GOTO 120
190 PRINT "MI":GOTO 120
200 PRINT "FA":GOTO 120
210 PRINT "SO":GOTO 120
220 PRINT "LA":GOTO 120
230 PRINT "SI":GOTO 120
```



UM NOVO TECLADO

O programa do Apple utiliza a rotina de produção de sons do artigo *Apple e TK-2000: Efeitos Sonoros*. O TK-2000 não precisa dela, já que dispõe do comando **SOUND**.

A linha 10 limpa o vídeo e ativa a tela gráfica de baixa resolução. A linha 20 transfere para a memória os códigos da rotina de geração de som, contida na linha 30.

A linha 35 cuida de colorir a tela, usando uma série de traços horizontais — verdes, no Apple — desenhados por comandos **HLIN**. As linhas 40 a 70 desenham um teclado de piano. As teclas brancas e pretas são feitas por comandos **VLIN**. As coordenadas para uso desses comandos ficam nas linhas **DATA** 50 e 70.

As linhas 80 e 90 imprimem as letras das teclas correspondentes ao desenho do piano, para orientar o usuário. Observe que há dois espaços entre cada par de letras. A linha 100, por sua vez, imprime o nome das notas (os tradicionais dó, ré, mi...).

A seguir, o programa espera que seja pressionada uma tecla, colocando seu caractere em **K$**.

Na linha 110, o comando **POKE 900,60** define a duração das notas — o endereço 900 é usado justamente para conter essa duração.

Depois disso, uma grande quantidade de linhas **IF...THEN** testa o conteúdo de **K$** para descobrir qual tecla foi pressionada, emitindo a nota correspondente em cada caso. O endereço 901 é usado para determinar a tonalidade. A rotina de produção de som é executada pelo comando **CALL 800**.

Detectada a tecla pressionada, o programa retorna à linha 110 para aguardar uma nova nota.



Por que as teclas musicais criadas pelo programa nos microcomputadores da linha TRS-Color não têm auto-repetição?

O programa do TRS-Color usa habitualmente o comando **INKEY$** para verificar o teclado.

É possível obtermos a auto-repetição das teclas por intermédio do comando **PEEK** (essa forma tem sido muito utilizada por nós em rotinas de movimentação do cursor). No entanto, o programa teria que acionar, neste caso, mais de vinte **PEEK** com diferentes números, um para cada tecla musical.

Como tais números não teriam nenhuma relação entre si (as teclas, neste caso, são QWE... e não ABC...), seria preciso colocá-los em linhas separadas ou armazená-los em uma linha **DATA**, tornando a execução do programa muito demorada e complicada. É mais rápido, cômodo e simples pressionar uma tecla de cada vez.

A duração de uma nota pode ser controlada pelo tempo em que a tecla se mantém sob pressão?

De um modo geral, a resposta é não. Na maioria dos computadores isso só seria possível com linguagem de máquina, já que o programa teria que fazer duas coisas ao mesmo tempo: tocar a nota e verificar se a tecla continua sendo pressionada.

Contudo, no micro MSX esse controle é possível; é que, nesse computador, o dispositivo de som funciona independentemente da unidade de processamento central.

A produção de acordes — emissão de mais de uma nota simultaneamente — para enriquecer uma melodia também é possível no MSX, que conta com três canais de som.

```
10  HOME : GR
20  FOR I = 0 TO 22: READ A: PO
KE 800 + I,A: NEXT
30  DATA  160,0,174,133,3,238,4
8,192,136,208,5,206,132,3,240,6
,202,208,245,76,34,3,96
35  COLOR= 12: FOR I = 0 TO 39:
 HLIN 0,39 AT I: NEXT
40  COLOR= 0: FOR I = 1 TO 7: R
EAD A: VLIN 20,30 AT A: NEXT
50  DATA  8,11,17,20,23,29,32
60  COLOR= 15: FOR I = 1 TO 20:
 READ A: VLIN 20,39 AT A: NEXT
70  DATA  6,7, 9,10, 12,13,15,1
6,18,19, 21,22,24,25,27,28,30,3
1,33,34
80  VTAB 21: HTAB 9: PRINT "2
3"; TAB( 18);"5   6   7"; TAB( 30
);"9   0"
90  VTAB 22: HTAB 8: PRINT "Q
W  E  R  T  Y  U  I  O  P"
100  VTAB 24: HTAB 7: PRINT "DO
 RE MI FA SO LA SI DO";
110  GET K$: POKE 900,60
120  IF K$ = "Q" THEN  POKE 901
,96: CALL 800: GOTO 110
130  IF K$ = "2" THEN  POKE 901
,90: CALL 800: GOTO 110
140  IF K$ = "W" THEN  POKE 901
,85: CALL 800: GOTO 110
150  IF K$ = "3" THEN  POKE 901
,80: CALL 800: GOTO 110
160  IF K$ = "E" THEN  POKE 901
,76: CALL 800: GOTO 110
170  IF K$ = "R" THEN  POKE 901
,72: CALL 800: GOTO 110
180  IF K$ = "5" THEN  POKE 901
,67: CALL 800: GOTO 110
190  IF K$ = "T" THEN  POKE 901
,64: CALL 800: GOTO 110
200  IF K$ = "6" THEN  POKE 901
,60: CALL 800: GOTO 110
210  IF K$ = "Y" THEN  POKE 901
,56: CALL 800: GOTO 110
220  IF K$ = "7" THEN  POKE 901
,53: CALL 800: GOTO 110
230  IF K$ = "U" THEN  POKE 901
,50: CALL 800: GOTO 110
240  IF K$ = "I" THEN  POKE 901
,47: CALL 800: GOTO 110
250  IF K$ = "9" THEN  POKE 901
,45: CALL 800: GOTO 110
260  IF K$ = "O" THEN  POKE 901
,42: CALL 800: GOTO 110
270  IF K$ = "0" THEN  POKE 901
,40: CALL 800: GOTO 110
280  IF K$ = "P" THEN  POKE 901
,37: CALL 800: GOTO 110
290  GOTO 110
```

O programa acima funciona normalmente nos micros da linha TK-2000. Contudo, muitos de seus usuários podem preferir usar o comando **SOUND**.

Para isto, basta eliminar as linhas 20 e 30 e modificar a linha 110, dando-lhe a seguinte forma:

```
110 GET K$: D = 60
```

A seguir, modifique todas as ocorrências de:

```
POKE 901,F:CALL 800
```

para

```
SOUND F,D
```

onde **F** não é uma variável, mas o mesmo número que ocorre após o **POKE 901**.

Os usuários do TK-2000 podem mudar as cores nas linhas 35, 40 e 60, já que as cores do Apple são diferentes.

O programa começa limpando a tela e colorindo-a de azul. A mesma linha também estabelece os parâmetros de volume, duração e velocidade das notas que serão executadas pelo comando **PLAY**. O artigo *Quebre a Barreira do Som* explica a sintaxe desse comando.

As linhas 20 e 30 posicionam as notas usadas pelo programa dentro da matriz **N$**. A linha 40 coloca num cordão as teclas que serão utilizadas e em outro os nomes correspondentes a algumas delas (dó, ré, mi etc).

As linhas 50 a 110 cuidam dos desenhos na tela, colorindo-os de laranja e preto. Depois disso, o computador espera que pressionemos uma tecla. Se esta não corresponder a uma nota, o computador voltará à linha 120 e esperará por outra.

A linha 140 executa a nota com o auxílio do comando **PLAY**. A tonalidade da nota é definida pela variável **NT**, que depende da tecla pressionada. **NT** é usada para selecionar o elemento da matriz **N$** que contém os parâmetros adequados da instrução **PLAY**.

Usando o comando **ON...GOTO**, o computador imprimirá na tela o nome da nota (se esta fizer parte da primeira oitava de " C maior"), retornando em seguida para esperar uma nova tecla.

```
10 CLS 3:PLAY"V31L4T8":B$=CHR$(
175)
20 DIM N$(21):FOR K=0 TO 19:REA
D N$(K):NEXT
30 DATA C,C#,D,D#,E,F,F#,G,G#,A
,A#,B,O4C,O4C#,O4D,O4D#,O4E,O4F
,O4F#,O4G
40 M$="Q2W3ER5T6Y7UI9O0P@-"+CHR
$(8):D$="QWERTYUIO"
50 FOR K=0 TO 6:READ A:N=2*A-20
*INT(A/10):POKE 1093+N,A:POKE 1
125+N,A:NEXT
60 DATA 50,51,53,54,55,57,48
65 POKE 1113,45:POKE 1145,45
70 FOR K=0 TO 11:READ A:POKE 11
24+K*2,A:POKE 1156+K*2,A:POKE 1
188+K*2,A:NEXT
80 DATA 81,87,69,82,84,89,85,73
,79,80,64,95
90 PRINT @260,"d"B$"r"B$"m"B$"f
"B$"s"B$"l"B$"s"B$"d";
100 PRINT @292,"o"B$"e"B$"i"B$"
a"B$"o"B$"a"B$"i"B$"o";
110 SCREEN 0,1
120 A$=INKEY$:IF A$="" THEN 120
130 NT=INSTR(M$,A$):IF NT=0 THE
N 120
140 PLAY"O3"+N$(NT-1)
150 PRINT @0,;:D=INSTR(D$,A$):O
N D GOTO 170,180,190,200,210,22
0,230,170
160 PRINT B$;B$;:GOTO 110
170 PRINT"DO";:GOTO 110
180 PRINT"RE";:GOTO 110
```

```
190 PRINT"MI";:GOTO 110
200 PRINT"FA";:GOTO 110
210 PRINT"SO";:GOTO 110
220 PRINT"LA";:GOTO 110
230 PRINT"SI";:GOTO 110
```

UMA SINFONIA DE BEETHOVEN

O que podemos tocar nesse teclado musical que o programa acaba de criar? Que tal uma canção de ninar, por exemplo? Ela começa com dó, de forma que a primeira tecla é Q; você terá que descobrir o ritmo "de ouvido":

Q,W,R,E,T,E,Q
Q,E,W,R,E,Q
Q,E,W,R,E,T,E,Q
Y,W,R,E,Q

E, agora, esta passagem famosa da *Nona Sinfonia* de Beethoven, composta quando o artista já estava surdo:

E,E,R,T,T,R,E,W
Q,Q,W,E,E,W,W
E,E,R,T,T,R,E,W
Q,Q,W,E,W,Q,Q
W,E,Q,W,E,R,E,Q
W,E,R,E,W,Q,W,T
E,E,R,T,T,R,E,W
Q,Q,W,E,W,Q,Q

Executemos a seguir o tema do filme *A Noviça Rebelde*:

Q,W,E,Q,E,Q,E
W,E,R,R,E,W,R
E,R,T,E,T,E,T
R,T,Y,Y,T,R,Y
T,Q,W,E,R,T,Y
Y,W,E,5,T,Y,U
U,E,5,6,Y,U,I
I,U,Y,R,U,T,I

Nessa melodia existem apenas três notas fora da escala (perto do final). Aqui estão as teclas para tocar o início de *Jesus Alegria dos Homens*, de Johann Sebastian Bach:

Q,W,E,T,R,R,Y,T,T,I,U,I,T,E,
Q,W,E,R,T,Y,T,R,E,W,E,Q

Infelizmente, neste ponto, a musica "cai fora do teclado".



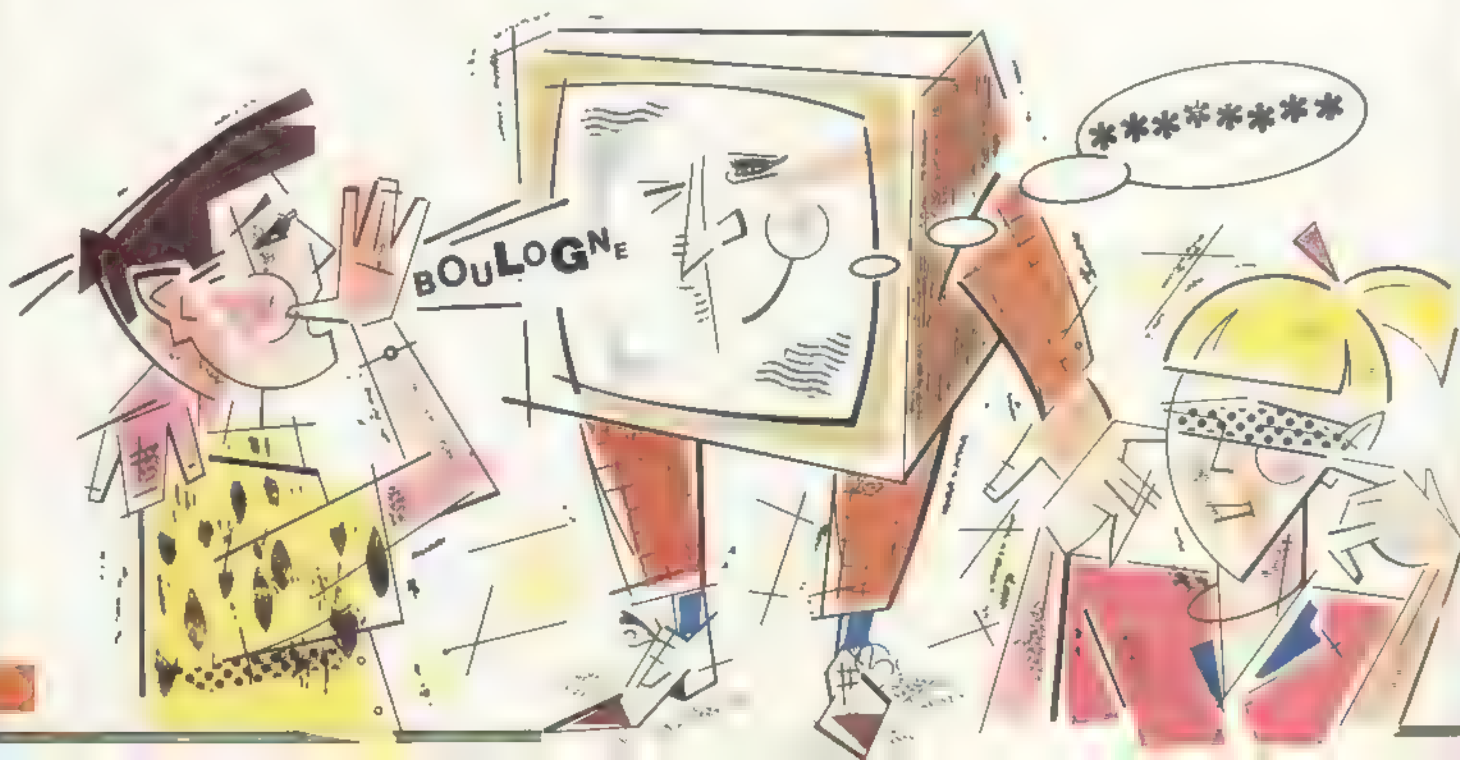
COMPLETE O JOGO DE PALAVRAS

Nonada. É com essa palavra misteriosa que tem início Grande Sertão: Veredas, de Guimarães Rosa. Surpreenda seus adversários com expressões como essa, jogando o jogo de palavras.

Neste artigo completaremos o jogo de palavras começado na lição anterior. Nele você encontrará tudo o que ainda é necessário para começar a jogar. Digite as linhas que faltam e veja algumas aplicações interessantes para os comandos de manipulação de cordões alfanuméricos do computador. Depois, desafie seus amigos para adivinhar palavras bem complicadas.

Aqui estão as rotinas para cada uma das opções do jogo: comprar letras, adivinhar uma letra em uma posição específica da frase, e adivinhar a frase inteira. O programa também faz a contagem de pontos assim como do número de tentativas e de jogadas.

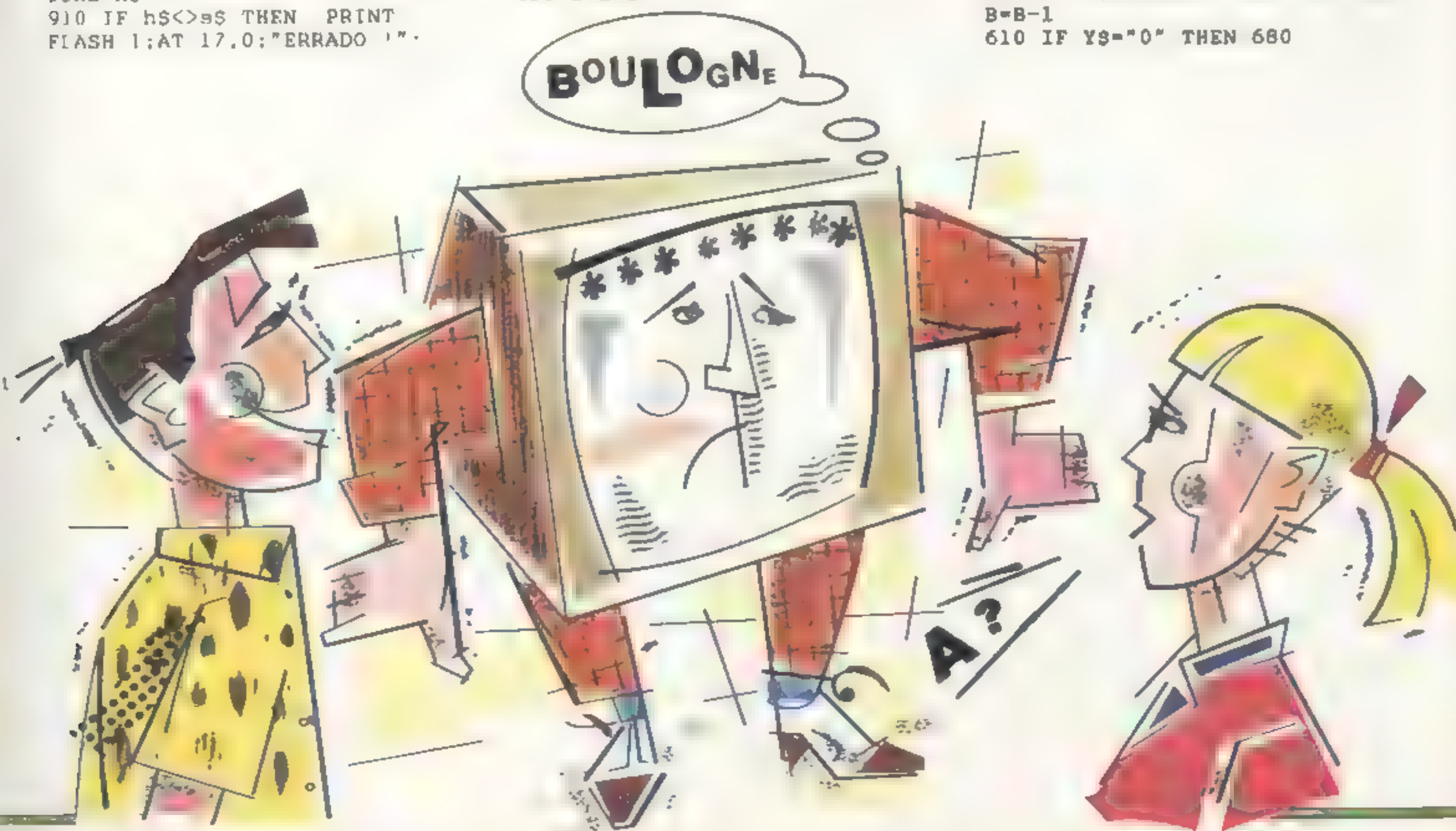
```
370 IF d$<>"XX" AND d$<>"ZZ"
AND LEN d$>1 THEN  GOTO 360
380 IF d$=CHR$ 32 THEN  GOTO
410
385 IF d$="ZZ" THEN  LET d$=""
: GOTO 900
390 IF CODE d$<65 OR CODE d$>
90 THEN  GOTO 360
400 IF d$="XX" THEN  LET d$=""
: GOTO 500
410 GOSUB 790
420 LET e=0
430 LET e=e+1
440 IF e=l+1 THEN  LET tb=tb-g
: LET q$(m TO m+7)="": PRINT
AT 4,0,q$: LET d$="": GOTO 470
450 IF s$(e)<>d$ THEN  GOTO
 430
460 IF s$(e)=d$ THEN  LET z$(e
)=d$: GOTO 430
470 PAUSE 100  PRINT AT 14,0;:
FOR r=1 TO 7  PRINT "
       "  NEXT r
480 PRINT  PAPER 2; INK 6:AT 1
,22;tb;CHR$ 32. PRINT  PAPER 2
, INK 6:AT 14,0,z$: PRINT " TE
NTATIVA ",f  LET f=f+1: IF s$=
z$ THEN  GOTO 730
490 GOTO 360
500 INPUT "QUAL CARACTER QUER
ADIVINHAR ?", LINE d$
510 IF LEN d$>1 THEN  GOTO 500
520 IF d$=CHR$ 32 THEN  GOSUB
790  GOTO 550
530 IF CODE d$<65 OR CODE d$>
90 THEN  GOTO 500
540 GOSUB 790
550 PRINT  PAPER 2; INK 6:AT
18,0;d$: PRINT AT 18,2;"EM QUA
L POSICAO ? - Use as       tecl
as  DIREITA e ESQUERDA       e
'0' para definir."
560 PRINT  PAPER 2; INK 6:AT
14,0;z$: PRINT  PAPER 6: INK 2
,AT 14,b,z$(b+1)
570 PAUSE 0  LET y$=INKEY$: IF
y$="" THEN  GOTO 570
590 IF y$="8" AND b<l 1 THEN
LET b=b+1
600 IF y$="5" AND b>0 THEN
LET b b 1
610 IF y$="0" THEN  GOTO 680
640 IF b>=32 THEN  LET w=15
LET v=b-32
650 IF b<32 THEN  LET w=14:
LET v b
660 PRINT  PAPER 2; INK 6:AT
14,0;z$: PRINT  PAPER 6: INK 2
;AT w,v;z$(b+1)
670 GOTO 570
680 IF z$(b+1)<>"*" THEN  GOTO
570
690 IF s$(b+1)<>d$ THEN  LET
tb=tb-g/2: PRINT  FLASH 1:AT
17,0;"QUE AZAR!"  PAUSE 50
LET b=0: GOTO 470
700 IF s$(b+1)=d$ THEN  PRINT
FLASH 1;AT 17,0;"MUITO BEM !":
PAUSE 50: LET z$(b+1)=d$: LET
tb=tb+g: LET b=0
710 IF s$=z$ THEN  GOTO 730
```



- NOVAS ROTINAS PARA CONCLUIR O JOGO DE PALAVRAS
- COMPRE LETRAS
- VERIFICAÇÃO DO PALPITE DO JOGADOR
- COMO ADIVINHAR UMA LETRA EM POSIÇÃO ESPECÍFICA
- ADIVINHE A FRASE COMPLETA
- A INSTRUÇÃO **CALL** NO APPLE E NO TK 2000

```
720 GOTO 470
730 PRINT  INK 6; PAPER 2:AT 1
.22,tb· PRINT AT 17.0:"PARABEN
S, ":b$;TAB 0;TAB 31;" "·
PAUSE 100· CLS
740 LET k=k+l· IF k=t*2 THEN
GOTO 880
750 LET c$=a$: LET a$=b$· LET
b$=c$
760 LET tc=ta: LET ta=tb: LET
tb=tc
770 LET q$=""· LET d=0· LET
f=1
780 GOTO 160
790 LET m=(CODE d$-64)*8-7
800 IF m= 263 THEN  LET m=209
810 IF q$(m TO m+5)="" THEN
GOTO 360
820 LET g=VAL q$(m+2 TO m+3)
830 RETURN
880 IF ta>tb THEN  CLS : PRINT
a$;" VENCEU POR ";ta;" A ";tb
890 IF tb>ta THEN  CLS · PRINT
b$;" VENCEU POR ",tb;" A ";ta
892 IF ta=tb THEN  CLS · PRINT
" O RESULTADO FOI UM EMPATE !"
895 STOP
900 INPUT "INTRODUZA A FRASE",
LINE h$
910 IF h$<>a$ THEN  PRINT
FLASH 1;AT 17.0:"ERRADO !"·
PAUSE 50· LET tb=tb-50· PRINT
INK 6: PAPER 2;AT 1,22;tb·
PRINT AT 15,0:"TENTATIVA ";f·
LET f=f+1: GOTO 360
920 FOR n=1 TO l· LET d$=z$(n)
· IF d$<>"*" THEN  GOTO 950
930 LET m=(CODE s$(n)-64)*8-7
IF m=-263 THEN  LET m=209
940 LET tb=tb+VAL q$(m+2 TO m+
3)
950 NEXT n  GOTO 730
```

```
360 PRINT @384,"";:LINE INPUT "
XX=ADIVINHAR LETRA
 ZZ=ADIVINHAR A FRASE
  A-Z=COMPRAR O CARACTER  ?";D$
370 IF D$<>"XX" AND D$<>"ZZ" AN
D LEN(D$)>1 THEN 360
380 IF D$=CHR$(32) THEN 410
385 IF D$="ZZ" THEN D$="":GOTO
1000
390 IF D$<"A" OR D$>"Z" THEN360
400 IF D$="XX" THEN D$="":GOTO
500
410 GOSUB 790
420 E=0
430 E=E+1
440 IF E=L+1 THEN TB=TB-G·MID$(
Q$,M,8)="        ":PRINT @96,Q$
:D$=""·GOTO 470
450 IF MID$(S$,E,1)<>D$ THEN430
460 IF MID$(S$,E,1)=D$ THEN MID
$(Z$,E,1)=D$·GOTO 430
470 GOSUB 950·PRINT @448·PRINT
@416:PRINT @480,STRING$(31,32);
480 PRINT @54,TB:PRINT @352,Z$·
PRINT @320,"TENTATIVA";F:F=F+1·
IF S$=Z$ THEN 730
490 GOTO 360
500 PRINT @448:PRINT @416:PRINT
 @448,"";:LINE INPUT "QUER ADIV
INHAR QUAL CARACTER ?";D$
510 IF LEN(D$)>1 THEN 500
520 IF D$=CHR$(32) THEN GOSUB 7
90:GOTO 550
530 IF D$<"A" OR D$>"Z" THEN500
540 GOSUB 790
550 PRINT @448,"EM QUAL POSICAO
? - USE SETAS E  '0' PARA DEFIN
IR.";
560 PRINT @352,Z$:POKE 1024+352
+B,(ASC(MID$(Z$,B+1,1))AND 191)
570 Y$=INKEY$:IF Y$="" THEN 570
590 IF Y$=CHR$(9) AND B<L-1 THE
N B=B+1
600 IF Y$=CHR$(8) AND B>0 THEN
B=B-1
610 IF Y$="0" THEN 680
```



```
660 PRINT @352,Z$:POKE 352+1024
+B,(ASC(MID$(Z$,B+1,1))AND 191)
670 GOTO 570
680 PRINT @480,STRING$(31,32);·
IF MID$(Z$,B+1,1)=D$ THEN TB=TB
-G·PRINT @448,"TRAPACEIRO!":FOR
 DE=1 TO 100:NEXT·B=0:GOTO 470
690 IF MID$(S$,B+1,1)<>D$ THEN
TB=TB-G/2·PRINT @448,"QUE AZAR!
":FOR DE=1 TO 100:NEXT:B=0:GOTO
 470
700 IF MID$(S$,B+1,1)=D$ THEN P
RINT @448,"MUITO BEM!":FOR DE=1
 TO 100:NEXT:MID$(Z$,B+1,1)=D$:
TB=TB+G:B=0
710 IF S$=Z$ THEN 730
720 GOTO 470
730 PRINT @480,"PARABENS, ";B$;
:GOSUB 950:CLS
740 K=K+1:IF K=T*2 THEN 880
750 C$=A$·A$=B$·B$=C$
760 TC=TA·TA=TB:TB=TC
770 Q$="":D=0:F=1
780 GOTO 160
790 M=(ASC(D$)-64)*8-7
800 IF M=-263 THEN M=209
810 IF MID$(Q$,M,6)="      "THE
N 360
820 G=VAL(MID$(Q$,M+2,2))
830 RETURN
880 IF TA>TB THEN CLS:PRINT A$;
" VENCEU POR";TA;"A";TB
890 IF TB>TA THEN CLS:PRINT B$;
" VENCEU POR";TB;"A";TA
892 IF TA=TB THEN CLS:PRINT"O R
ESULTADO FOI UM EMPATE !"
895 END
1000 PRINT @448:PRINT @416:PRIN
T @416,"INTRODUZA A FRASE -":LI
NE INPUT GU$
1010 IF GU$<>S$ THEN PRINT @416
,"ERRADO !":TB=TB-50·PRINT @54,
TB·GOSUB 950·PRINT @320,"TENTAT
IVA";F::F=F+1·GOTO 360
1020 FOR N=1 TO L:D$=MID$(Z$,N,
1):IF D$<>"*" THEN 1050
1030 M=(ASC(MID$(S$,N,1))-64)*8
-7:IF M=-263 THEN M=209
1040 TB=TB+VAL(MID$(Q$,M+2,2))
1050 NEXT N:GOTO 730
```



```
167 S$=""
360 LOCATE 0,21·PRINTSPC(77)·LO
CATE 0,21·PRINT"XX-Adivinha let
ra     ZZ-Adivinha frase"·INPUT"
          A-Z Compra letra ",D$
370 IF D$<>"XX" AND D$<>"ZZ" AN
D LEN(D$)>1 THEN 360
380 IF D$="" THEN D$="_"·GOTO 4
10
385 IF D$="ZZ" THEN D$=""·GOTO
1000
390 IF (D$<"A" OR D$>"Z") AND D
$<>"Ç" THEN 360
400 IF D$="XX' THEN D$="" GOTO
500
410 GOSUB 790
420 E=0
430 E=E+1
440 IF E=L+1 THEN TB=TB-G·MID$(
Q$,M,7+(N/5=INT(N/5)))=STRING$(
8,32)·LOCATE 0,4·PRINTQ$ D$ ""·
GOTO 470
450 IF MID$(S$,E,1)<>D$ THEN 43
0
460 IF MID$(S$,E,1)=D$ THEN MID
$(Z$,E,1)=D$·GOTO 430
470 GOSUB 950
480 LOCATE25,1:PRINTTB;"pontos"
:LOCATE 0,12:PRINTZ$:LOCATE 0,1
5·PRINT"Tentativa ";F·F=F+1·IF
S$=Z$ THEN 730
490 GOTO 360
500 LOCATE 0,21·PRINTSPC(77)·LO
CATE 0,21·INPUT"QUAL A LETRA";D
$
510 IF LEN(D$)>1 THEN 500
520 IF D$="" THEN D$=CHR$(32):G
OSUB 790:GOTO 550
530 IF (D$<"A" OR D$>"Z") AND D
$<>"Ç" THEN 500
540 GOSUB 790
550 LOCATE 0,21.PRINTSPC(77)·LO
CATE 0,21.PRINT"INDIQUE A POSIÇ
AO USANDO AS SETAS          PARA
MARCAR TECLE <0>"
555 B=0·V=12·BB=0
560 LOCATE BB,V,1
570 Y$=INKEY$·IF Y$="" THEN 570
590 IF Y$=CHR$(28) AND B<L-1 TH
EN B=B+1
600 IF Y$=CHR$(29) AND B>0 THEN
 B=B-1
605 BB=B:V=12·IF B>39 THEN BB=B
-40 V=13LIST 600
610 IF Y$="0" THEN 680
670 GOTO 560
680 IF MID$(Z$,B+1,1)=D$ THEN T
B=TB G·LOCATE 0,21·PRINTSPC(77)
:LOCATE 0,21:PRINT"TRAPACEIRO!"
:GOSUB 950:GOTO 470
690 IF MID$(S$,B+1,1)<>D$ THEN
TB=TB-G/2·LOCATE 0,21·PRINTSPC(
77)·LOCATE 0,21 PRINT"ERROU DES
TA VEZ   "·GOSUB 950·GOTO 470
700 IF MID$(S$,B+1,1)=D$ THEN L
OCATE 0,21·PRINTSPC(77):LOCATE
0,21·PRINT"BOM PALPITE." MID$(Z
$,B+1,1)=D$·GOSUB 950·TB=TB+G
710 IF S$ Z$ THEN 730
720 GOTO 470
730 LOCATE 0,21·PRINTSPC(77)·LO
CATE 0,21 PRINT"PARABENS! ACERT
OU!"·BEEP·GOSUB 950 BEEP·CLS
740 K=K+1·IF K=T*2 THEN 880
750 SWAP A$,B$
760 SWAP TA,TB
770 Q$="":D=0 F=1·S$=""
780 GOTO 160
790 M=INSTR(Q$,D$)
800 IF D$ "_" THEN D$=CHR$(32)
810 IF MID$(Q$,M,5)=STRING$(5,3
2) THEN 360
```



```
820 G=VAL(MID$(Q$,M+2,2))
830 RETURN
880 IF TA>TB THEN CLS:PRINTA$;" GANHOU POR";TA;"PONTOS A";TB
890 IF TB>TA THEN CLS:PRINTB$;" GANHOU POR",TB;"PONTOS A";TA
892 IF TA=TB THEN CLS:PRINT"O JOGO TERMINOU EMPATADO EM":PRINT TA;"PONTOS"
895 END
950 FOR DE=1 TO 1000:NEXT:RETURN
1000 LOCATE 0,21:PRINTSPC(77):LOCATE 0,21:PRINT"DIGITE A FRASE COMPLETA ".LINEINPUT GU$
1010 IF GU$<>S$ THEN LOCATE 0,21:PRINTSPC(7):LOCATE 0,21:PRINT"ERRADO!".TB=TB-50:LOCATE 25,1:PRINTTB,"pontos":GOSUB 950:LOCATE 0,15:PRINT"TENTATIVA";F:F=F+1 GOTO 360
1020 FOR N=1 TO L:D$=MID$(Z$,N,1):IF D$<>"*" THEN 1050
1030 IF D$=CHR$(32) THEN D$="_":M=INSTR(Q$,D$)
1040 TB=TB+VAL(MID$(Q$,M+2,2))
1050 NEXT:GOTO 730
```



```
370 IF D$ < > "ZZ" AND D$ < > "XX" AND LEN (D$) > 1 THEN 360
380 IF D$ = "" THEN D$ = CHR$ (32): GOTO 410
385 IF D$ = "ZZ" THEN D$ = "": GOTO 1000
390 IF D$ < "A" OR D$ > "Z" THEN 360
400 IF D$ = "XX" THEN D$ = "": GOTO 500
410 GOSUB 790
420 E = 0
430 E = E + 1
440 IF E = L + 1 THEN TB = TB - G:Q$ = LEFT$ (Q$,M - 1) + "  " + MID$ (Q$,M + 8): VTAB 5: PRINT Q$:D$ = "": GOTO 470
450 IF MID$ (S$,E,1) < > D$ THEN 430
460 IF MID$ (S$,E,1) = D$ THEN Z$ = LEFT$ (Z$,E) + D$ + MID$ (Z$,E + 2): GOTO 430
470 FOR DE = 1 TO 300: NEXT
480 VTAB 2: HTAB 25: PRINT TB;" PONTOS ": VTAB 12: HTAB 40: PRINT Z$: VTAB 16: PRINT "TENTATIVA ";F:F = F + 1: IF S$ = MID$ (Z$,2) THEN 710
490 GOTO 360
500 VTAB 22: CALL - 958: INPUT "QUAL A LETRA ";D$
510 IF LEN (D$) > 1 THEN 500
520 IF D$ = "" THEN D$ = " ": GOSUB 790: GOTO 550
530 IF D$ < "A" OR D$ > "Z" THEN 500
540 GOSUB 790
550 VTAB 22: CALL - 958: PRINT "COLOQUE O MARCADOR SOB A POSICAO DESEJADA E TECLE [ENTER]";:B = 1
560 VTAB 14: CALL - 868: HTAB B: PRINT CHR$ (94)
570 GET Y$: IF Y$ = CHR$ (13) THEN 680
590 IF Y$ = CHR$ (8) AND B > 1 THEN B = B - 1
600 IF Y$ = CHR$ (21) AND B < L THEN B = B + 1
670 GOTO 560
680 VTAB 14: CALL - 958: VTAB 22: IF MID$ (Z$,B,1) = D$ THEN PRINT "TRAPACEIRO!";:TB = TB - G: FOR DE = 1 TO 300: NEXT : GOTO 470
690 IF MID$ (S$,B,1) < > D$ THEN TB = TB - G / 2: PRINT "ERROU DESTA VEZ...";: FOR DE = 1 TO 300: NEXT : GOTO 470
700 IF MID$ (S$,B,1) = D$ THEN PRINT "BOA! ";:TB = TB + G:Z$ = LEFT$ (Z$,B) + D$ + MID$ (Z$,B + 2): FOR DE = 1 TO 300: NEXT
710 IF S$ = MID$ (Z$,2) THEN FLASH : VTAB 13: PRINT S$: CALL - 868: NORMAL : GOTO 730
720 GOTO 470
730 VTAB 22: CALL - 958: PRINT "PARABENS!": GOSUB 950: HOME
740 J = J + 1: IF J = T * 2 THEN 880
750 C$ = A$:A$ = B$:B$ = C$
760 TC = TA:TA = TB:TB = TC
770 Q$ = "":D = 0:F = 1
780 GOTO 160
790 M = ( ASC (D$) - 64) * 8 - 6
800 IF D$ = CHR$ (32) THEN M = 218
805 IF D$ = "Z" THEN M = 210
810 IF MID$ (Q$,M,2) = "  " THEN POP : GOTO 360
820 G = VAL ( MID$ (Q$,M + 2,2))
830 RETURN
880 IF TA > TB THEN HOME : PRINT : PRINT A$;" GANHOU POR ";TA;" PONTOS A ";TB
890 IF TA < TB THEN HOME : PRINT : PRINT B$;" GANHOU POR ";TB;" PONTOS A ";TA
892 IF TA = TB THEN HOME : PRINT : PRINT "O JOGO TERMINOU EMPATADO EM ": PRINT TA;" PONTOS!"
895 END
950 FOR DE = 1 TO 3000: NEXT : RETURN
1000 VTAB 22: CALL - 958: INPUT "ENTRE A FRASE: ";GU$
1010 IF GU$ < > S$ THEN VTAB 22: CALL - 958: PRINT "ERRADO
```



Por que as linhas em que há a instrução CALL diferem nos programas do Apple e do TK-2000?

Porque a instrução **CALL** seguida de um certo número de linhas serve para acionar uma rotina em linguagem de máquina. No nosso caso, essas linhas constituem rotinas intrínsecas do computador, ou seja, são rotinas próprias da máquina

No Apple, a instrução **CALL −958** faz com que a tela seja apagada desde o cursor até a última posição de vídeo, sem alteração na posição do cursor. No TK-2000, ela faz a mesma coisa, mas com uma diferença: agora, o cursor é colocado na primeira posição da tela. Assim, ele deve ser reposicionado após o comando

Já a instrução **CALL −868** provoca, no Apple, o apagamento da linha em que está o cursor a partir da posição deste. A mesma instrução no TK-2000 não tem um efeito necessário para o nosso programa

```
!":TB = TB - 50: VTAB 2: HTAB 2
5: PRINT TB;" PONTOS": VTAB 16:
 PRINT "TENTATIVA ";F:F = F + 1
: FOR DE = 1 TO 300: NEXT : GOT
O 360
1020  FOR N = 2 TO L + 1:DS =
MIDS (ZS,N,1): IF DS <  > "*" T
HEN 1050
1030 M = ( ASC ( MIDS (SS,N - 1
,1)) - 64) * 8   6· IF M =   2
62 THEN M = 218
1035  IF M = 202 THEN M = M + B

1040 TB = TB +  VAL ( MIDS (QS.
M + 2,2))
1050  NEXT ·ZS = " " + SS· GOTO
 710
```

Para executar o programa no TK-2000, faça as seguintes modificações:

```
500  VTAB 22· CALL  - 958· VTAB
 22: INPUT "QUAL A LETRA ";DS
550  VTAB 22: CALL  - 958: VTAB
 22· PRINT "COLOQUE O MARCADOR
SOB A POSICAO DESEJADA E TECLE
[ENTER]";·B = 1
560  VTAB 14  PRINT  SPC( 40)
VTAB 14· HTAB B· PRINT  CHRS (9
4)
680  VTAB 14: CALL  - 958: VTAB
 22· IF  MIDS (ZS,B,1) = DS THE
N  PRINT "TRAPACEIRO!";:TB = TB
 - G: FOR DE = 1 TO 300: NEXT :
 GOTO 470
710  IF SS =  MIDS (ZS,2) THEN
 VTAB 13· PRINT SS· GOTO 730
730  VTAB 22: CALL  - 958: VTAB
 22· PRINT "PARABENS!": GOSUB 9
50  HOME
1000  VTAB 22. CALL  - 958· VTA
B 22: INPUT "ENTRE A FRASE: ";G
US
1010  IF GUS <  > SS THEN  VTAB
 22: CALL  - 958: VTAB 22: PRIN
T "ERRADO!"·TB = TB - 50: VTAB
2  HTAB 25: PRINT TB;" PONTOS"·
 VTAB 16: PRINT "TENTATIVA ";F·
F = F + 1  FOR DE = 1 TO 300· N
EXT · GOTO 360
```

Como na parte anterior da listagem, existem pequenas variações entre as máquinas, mas, de maneira geral, os programas são muito parecidos.

As linhas 370 a 410 manipulam a opção do jogador — comprar letras, adivinhar letras ou mesmo descobrir a frase toda. A rotina identifica o tipo de palpite e evita que sejam feitas entradas ilegais.

COMO COMPRAR LETRAS

Se o jogador decidir comprar uma letra, o computador verificará imediatamente qual o valor dessa letra. Para isso, é acionada uma sub-rotina que começa na linha 790. Esta encontra o valor ASCII da letra para saber até que posição da tabela de valores deve ir. Se o computador encontrar um espaço em branco na posição, isso significa que a letra já foi comprada e o programa voltará à linha 360. Do contrário, será lido o valor da letra, por intermédio da função **VAL**. Esse valor é utilizado para calcular o novo número de pontos do jogador.

A rotina que cuida da compra de letras começa na linha 430. As linhas 430 a 460 passam pela frase procurando ocorrências da letra. A cadeia de aste-



riscos é atualizada, substituindo-se os asteriscos pela letra comprada, quando isso ocorrer. A nova cadeia é então mostrada na tela e o valor da letra subtraído do total de pontos. O número de tentativas é incrementado de 1. A linha 440 apaga a letra escolhida da tabela, indicando que ela não está mais disponível.

ADIVINHE LETRAS EM UMA POSIÇÃO

Se o jogador quiser tentar acertar uma letra em posição específica, deve primeiro selecionar XX. Esse procedimento faz o programa pular para a linha 500. O jogador deve então dizer que letra vai ser usada. Várias verificações evitam que se faça uma entrada ilegal. Em seguida, é preciso mover o marcador para a posição desejada e pressionar 0 (no Apple, pressione <ENTER>). O programa verifica se a letra está disponível e faz um teste para ver se o palpite é correto.

Caso o jogador erre o palpite, a linha 690 enviará uma mensagem, subtraindo metade do valor da letra do total de pontos. Se letra e posição coincidirem, a linha 700 emitirá uma mensagem de felicitações e somará o valor da letra ao total de pontos do jogador. Quando a frase é completada, a linha 710 envia o programa para a 730, que dá a boa notícia ao jogador.

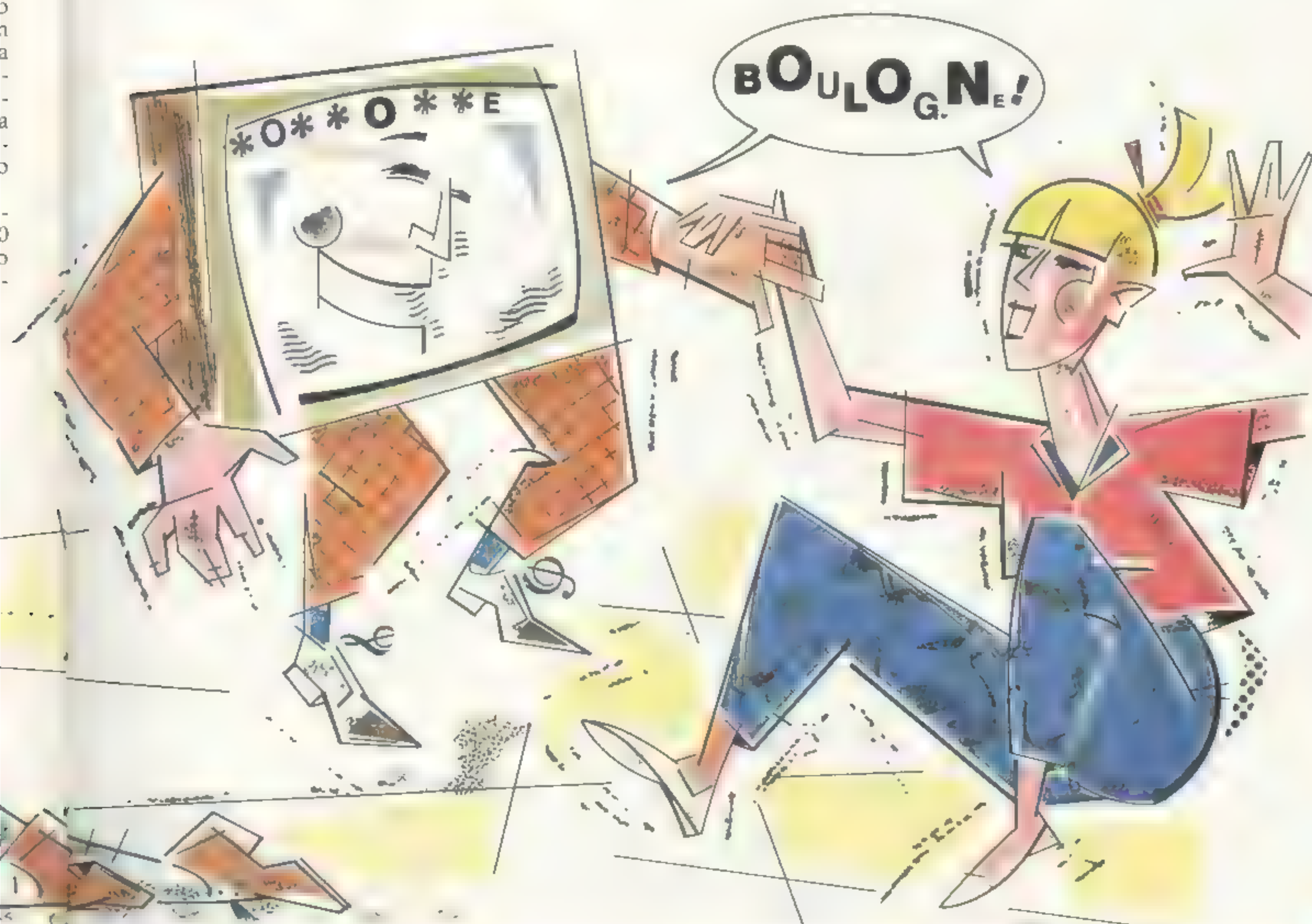
A FRASE COMPLETA

Se o jogador for mais ambicioso, pode querer adivinhar a frase inteira. Para isso, deve primeiro digitar ZZ. A linha 385 manda o programa para a linha 900 (no Spectrum) ou 1000 (nos outros micros). A rotina pede que a frase seja digitada e a compara com a original. Se elas não forem iguais, o jogador perderá cinqüenta pontos e o contador de tentativas será incrementado. Se o palpite for correto, o valor de todas as letras ainda não adivinhadas será somado ao total de pontos, e a linha 730 avisará o jogador que a frase está correta.

FIM DO JOGO

Adivinhada a frase, o programa verifica o número de jogadas realizadas. Este não pode ser maior que o determinado no início do jogo. Se o número ainda for menor, o próximo jogador será chamado, depois que uma nova frase tenha sido introduzida. Quando o jogo acaba, o programa passa à linha 880. Os pontos são comparados e o resultado final é apresentado.

O jogo termina aqui, mas você pode implementar uma rotina do tipo "joga novamente?" para deixá-lo completo.



SÍMBOLOS GRÁFICOS NO TK-2000

Os computadores da linha TK-2000 dispõem de um grande número de caracteres gráficos que podem ser entrados diretamente pelo teclado, ou usados em um programa.

As telas gráficas no computador (GR e HGR) podem ser programadas por meio de diversas instruções extremamente poderosas, em média e alta resolução, tais como **VLIN**, **HLIN**, **PLOT**, **HPLOT**, **DRAW** e outras.

Entretanto, o TK-2000 tem um recurso gráfico adicional, habitualmente pouco explorado, que os compatíveis com a linha Apple não têm. Trata-se dos *caracteres gráficos*, que estão disponíveis para o programador através de dois meios: entrada direta pelo teclado e inserção por intermédio da função **CHR$**, do BASIC.

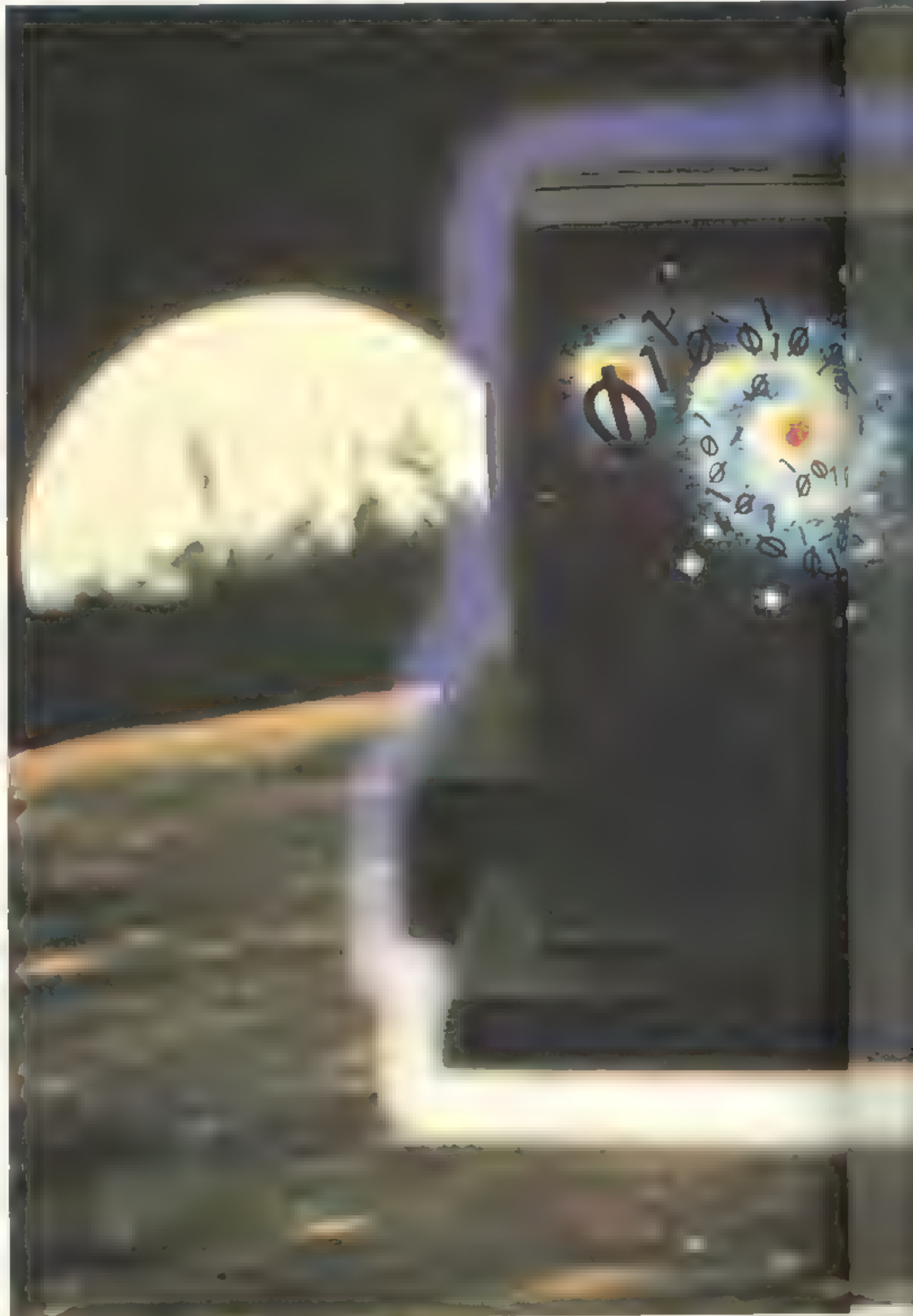
O que são caracteres gráficos? Como você já sabe, os caracteres que aparecem no vídeo têm códigos numéricos inteiros, que teoricamente podem variar entre 0 e 255. Cada caractere corresponde, portanto, a um byte da memória de vídeo. Parte dessa codificação, convencionada internacionalmente, é o chamado código ASCII, que vai de 32 a 126. Os códigos de 0 a 31 são normalmente utilizados em funções de controle do vídeo e dependem do tipo de computador que está sendo usado.

O mesmo acontece com os códigos que vão de 127 a 255. Nessa faixa, os fabricantes utilizam geralmente os códigos para acomodar caracteres gráficos (que podem ser tipos especiais, como naipes de baralho, notas musicais etc., ou blocos gráficos formando linhas, ângulos e cantos).

GRÁFICOS DA ROM

Tais caracteres gráficos são também chamados de *gráficos da ROM*, pois já vêm pré-programados. Nos computadores da linha TK-2000, os caracteres especiais ocupam a faixa da tabela de caracteres que vai de 193 a 242.

Os sinais gráficos que mais nos interessarão neste artigo são os blocos geralmente utilizados na composição de desenhos formados por linhas retas, tais como tabelas ou formulários de entrada. A vantagem desses blocos consiste em simplificar a programação, tornando desnecessária a mistura da tela gráfica com o texto. Tal simplificação deixa aberto o caminho para o emprego de



- SÍMBOLOS GRÁFICOS
- GRÁFICOS DA ROM
- COMO ENTRAR GRÁFICOS PELO TECLADO
- SIMPLIFIQUE A PROGRAMAÇÃO
- CARACTERES GRÁFICOS EM PROGRAMAS
- A FUNÇÃO CHR$
- OS NAIPES DO BARALHO
- TABELA DE REFERÊNCIA

comandos mais diretos, como **VTAB**, **HTAB**, **PRINT**, **INPUT** etc.

ENTRADA PELO TECLADO

Da mesma maneira que os caracteres convencionais do código ASCII (demarcados sobre as teclas do microcomputador), os sinais gráficos podem ser digitados pelo teclado. Para isso, é necessário, em primeiro lugar, pressionar simultaneamente as teclas **<CONTROL>** e **<B>**. Esse procedimento coloca o teclado em modo gráfico.

Em seguida, deve-se acionar simultaneamente a tecla **<SHIFT>**, e uma das teclas alfanuméricas, ou então as teclas **<SHIFT>**, **<CONTROL>** e uma terceira. Com isso, o caractere desejado aparece diretamente na tela (por exemplo, dentro de uma cadeia alfanumérica).

O manual de programação do TK-2000 exibe um desenho esquemático do teclado, no qual são assinalados todos os caracteres gráficos, em relação à disposição das teclas. Consulte também a tabela que apresentamos a seguir.

Ao se terminar de digitar os caracteres gráficos em uma linha, deve-se pressionar novamente **<CONTROL>** e **<B>**, para sair do modo gráfico.

Por exemplo, se quisermos digitar o símbolo tradicional do naipe de paus, devemos teclar **<CONTROL><B>** e, em seguida, **<SHIFT><R>**.

O programa abaixo demonstra como isso pode ser feito. Ele desenha uma tabela simples na tela, utilizando blocos gráficos e coloca depois os diversos nomes digitados nas linhas da tabela.

```
200 HOME
210 PRINT "┌──────┬─────────────┐"
220 PRINT "│ NO.  │    NOME     │"
230 PRINT "╞══════╪═════════════╡"
240 FOR I=1 TO 10
250 PRINT "│      │             │"
260 NEXT I
270 PRINT "└──────┴─────────────┘"
275 FOR I=1 TO 10
280 VTAB 20·HTAB 1
290 INPUT "NOME : ";N$
300 VTAB I+3:HTAB 4· PRINT I
310 VTAB I+3·HTAB 10: PRINT N$
320 NEXT I
330 VTAB 20:HTAB 1:STOP
```

Os traços devem ser digitados nesta sequência:

Linha 210
<SHIFT> A
<SHIFT><CONTROL> F (4 vezes)
<SHIFT> H
<SHIFT><CONTROL> F (11 vezes)
<SHIFT> S
Linha 220
<SHIFT><CONTROL> C
Linha 230
<SHIFT><CONTROL> H
<SHIFT><CONTROL> G (4 vezes)
<SHIFT><CONTROL> B
<SHIFT><CONTROL> G (11 vezes)
<SHIFT><CONTROL> N
Linha 250: como a linha 220
Linha 270
<SHIFT> Z
<SHIFT><CONTROL> F (4 vezes)
<SHIFT> G
<SHIFT><CONTROL> F (11 vezes)
<SHIFT> X

Tanto na linha 220 quanto na linha 250, os espaços em branco e as letras podem ser digitados normalmente, sem precisar sair do modo gráfico. Em outras palavras: nas linhas que têm caracteres gráficos (210, 220, 230, 250 e 270), basta pressionar **<CONTROL><B>** uma vez, logo após digitar o sinal de abre aspas; em seguida, teclar os gráficos e/ou as letras, e novamente **<CONTROL><B>**. Só depois disso, digita-se o sinal de fecha aspas.

Existem duas desvantagens nessa forma de entrada de caracteres gráficos: primeiro, as teclas não têm qualquer marcação que auxilie o usuário a encontrar o gráfico correto. Torna-se necessário então consultar o manual, o que faz o processo bastante moroso. Em segundo lugar, o programa não pode ser listado em uma impressora não gráfica, ou que não seja específica para a linha TK-2000.

CARACTERES GRÁFICOS NO PROGRAMA

Existe ainda um outro truque para especificar caracteres gráficos dentro de um programa sem precisar digitá-los diretamente. A função que permite fazer isso é a utilíssima **CHR$**.

Os caracteres normais, especiais e gráficos com códigos na faixa de 32 a 255 podem ser impressos na tela a par-

tir de um programa, contendo a **CHR$** acompanhada do número de código do caractere desejado. Por exemplo, para imprimir na tela o símbolo de paus (naipe de baralho), digitamos:

```
PRINT CHR$(242);CHR$(231)
```

Portanto, para combinar códigos gráficos com a função **CHR$**, é preciso "avisar" o computador que o código será usado como gráfico. Para isso, recorremos a dois bytes: **CHR$(242)**, seguido de **CHR$(n)**, onde **n** é o código do caractere gráfico na tabela. O próximo programa mostra a tabela de correspondência entre códigos numéricos e gráficos na tela do TK-2000:

```
10 HOME
20 FOR J=193 TO 242 STEP 6
30 FOR I=J TO J+5
40 PRINT I;" ";CHR$(242);CHR$(
I);" ";
50 NEXT I
60 PRINT
70 NEXT J
80 VTAB 20:HTAB 1: PRINT
"PRESSIONE RETURN"
90 GET A$
```

Os caracteres são impressos ordenadamente em fileiras, por meio dos dois laços que começam nas linhas 20 e 30. O primeiro laço varia **J** de 193 a 242 (faixa de códigos correspondente aos caracteres gráficos), de 6 em 6. O laço seguinte percorre todos os valores entre **J** e **J+5**. O **PRINT** da linha 60 serve para encerrar uma fileira de seis códigos e suas representações gráficas, que são mostradas na linha 40.

CARACTERES GRÁFICOS PARA O TK-2000

Código	Teclas	Caractere
193	CONTROL SHIFT 1	
194	CONTROL SHIFT 2	
195	CONTROL SHIFT 3	
196	CONTROL SHIFT 4	
197	CONTROL SHIFT 5	
198	CONTROL SHIFT 6	
199	CONTROL SHIFT 7	
200	CONTROL SHIFT Q	
201	CONTROL SHIFT W	
202	CONTROL SHIFT E	
203	CONTROL SHIFT R	
204	CONTROL SHIFT T	
205	CONTROL SHIFT Y	
206	CONTROL SHIFT U	
208	CONTROL SHIFT G	
209	CONTROL SHIFT H	
210	CONTROL SHIFT B	
211	CONTROL SHIFT N	
212	CONTROL SHIFT A	
213	CONTROL SHIFT S	
214	CONTROL SHIFT Z	
215	CONTROL SHIFT X	
216	CONTROL SHIFT D	
217	CONTROL SHIFT F	
218	CONTROL SHIFT V	
219	CONTROL SHIFT C	
220	CONTROL SHIFT J	
221	CONTROL SHIFT M	
222	SHIFT T	●
223	SHIFT J	
224	SHIFT G	
225	SHIFT H	
226	SHIFT B	
227	SHIFT N	
228	SHIFT Q	♠
229	SHIFT W	♥
230	SHIFT E	♦
231	SHIFT R	♣
232	SHIFT D	
233	SHIFT F	
234	SHIFT C	
235	SHIFT V	
236	SHIFT A	
237	SHIFT S	
238	SHIFT Z	
239	SHIFT X	
240	SHIFT U	
241	SHIFT Y	
242	SHIFT M	



Se quiser usar caracteres gráficos com certa frequência em um programa, deve armazenar o código de controle em uma variável alfanumérica, como no exemplo abaixo. Neste caso, os quatro símbolos dos naipes do baralho são armazenados em **N$** e seus nomes, em **E$**:

```
5 HOME
10 FOR I=1 TO 4
20 READ E$(I)
30 LET N$(I)=CHR$(242)+
CHR$(227+I)
40 PRINT N$(I),E$(I)
50 NEXT I
60 DATA ESPADAS,COPAS,OUROS,
PAUS
```

Assim, toda vez que precisarmos imprimir na tela um dos quatro símbolos, digitaremos **PRINT N$(N)**, onde **N** é o código do naipe (1 = espadas, 2 = copas, 3 = ouros e 4 = paus).

Do mesmo modo quando quisermos utilizar uma cadeia de caracteres gráficos em vários pontos de um programa, devemos armazená-lo em uma variável alfanumérica. Como o TK-2000 não tem a função **STRING$**, que permite fazer isso com um único comando, é preciso escrever um laço de acumulação:

```
10 HOME
20 INPUT "CODIGO GRAFICO
(193-242) : ";        C
30 INPUT "COMPRIMENTO (1-39) :
";L
40 PRINT:PRINT:PRINT
50 LET S$=""
60 FOR I=1 TO L
70 LET S$=S$+CHR$(242)+CHR$(C)
80 NEXT I
90 PRINT S$
100 GOTO 10
```

A cadeia alfanumérica **S$** tem **L** vezes dois códigos, pois é necessário entrar um código 242 antes de cada código gráfico C (não funciona colocar apenas um código 242 como primeiro caractere de **S$**). Se rodar esse programa com vários códigos ao acaso — ou entrando-os em seqüência para ver o resultado —, constatará que às vezes surge uma interferência na tela inteira. Se isso ocorrer, experimente interromper o programa, usando **<CONTROL> <C>** e digitando **RUN** novamente.

MAIS TÉCNICAS DE ORDENAÇÃO

Veja como pequenos melhoramentos nas rotinas de ordenação mais comuns podem aumentar sua velocidade de operação. E conheça uma rotina que bate todas as outras.

Todas as técnicas usadas para colocar dados em ordem apontam falhas, seja por dispenderem muito tempo, seja por exigirem um grande espaço de memória. Dessa forma, em cada aplicação específica é preciso procurar o método de ordenação mais eficiente, pois não existe uma técnica ideal que responda a todas as necessidades.

Já estudamos alguns dos métodos mais populares de ordenação de dados; a seguir, mostraremos outros, com a ressalva de que alguns deles são apenas refinamentos dos que foram apresentados anteriormente. Como você verá, estes são bem mais eficientes do que as versões originais.

SUBSTITUIÇÃO RETARDADA

O principal defeito da ordenação tipo bolha — a lentidão — torna-se particularmente evidente quando é preciso organizar uma grande quantidade de itens. No entanto, com uma pequena modificação no algoritmo original, pode-se reduzir o tempo gasto praticamente à metade.

A ordenação tipo bolha usa boa parte do tempo em comparar os diversos números da linha, trocando-os de posição até encontrar um valor maior. Nesse processo, várias trocas são feitas com grande dispêndio de tempo.

A rotina de substituição retardada funciona de modo semelhante, mas difere no fato de que nenhuma troca é realizada até que toda a linha tenha sido comparada. Vejamos um exemplo que emprega uma seqüência similar àquela estudada no artigo *Ordenação pelo Método de Bolhas* (página 292):

```
início                         fim

67   35   72   19   47   38   11   86
```

O primeiro valor, 67, é comparado (mas não trocado) com 35, como aconteceria na rotina de bolha normal; o mesmo acontece em relação ao primeiro número maior que ele, 72. Este último é tomado como o novo maior número e comparado sucessivamente com 19, 47, 38 e 11, antes de encontrar o valor máximo da seqüência: 86. Este permanece na última posição. O quadro que se segue mostra como tudo acontece, passo a passo. Alguns números aparecem entre colchetes. O primeiro deles é 86, o maior valor da primeira seqüência. Nas próximas linhas, os colchetes assinalam o maior número incorretamente localizado e o valor que ocupa seu lugar. Esses números são trocados na linha seguinte. Assim, na segunda linha, 72 é identificado como o maior incorretamente posicionado, e trocado com o 11, e assim por diante.

```
início                          fim

 67   35   72   19   47   38   11[86]
 67   35  [72]  19   47   38  [11] 86
[67]  35   11   19   47  [38]  72  86
 38   35   11   19  [47]  67   72  86
[38]  35   11  [19]  47   67   72  86
 19  [35][11]   38   47   67   72  86
[19][11]  35    38   47   67   72  86
 11   19   35   38   47   67   72  86
```

O número de comparações feito é o mesmo da rotina tipo bolha, mas o de trocas é muito menor.

Uma parte do programa a seguir é igual a um dos programas que aparecem no artigo *Rotinas de Ordenação*. Para executá-lo, basta adicionar as linhas de 4000 em diante (cuidado para não confundir a letra I com o número 1, ao fazer a digitação). Os usuários do Apple e do MSX devem fazer as modificações no programa principal conforme o indicado para todas as máquinas.



S

```
 10 POKE 23658,8: LET T=0:
 INPUT "NUMERO DE ITENS ";AA:
 IF AA<2 THEN  GOTO 10
 15 DIM A(AA)
 20 PRINT : PRINT "TABELA DESO
 RDENADA": PRINT
 30 FOR Z=1 TO AA
 40 LET A(Z)=INT (RND*100)+1
 50 PRINT TAB T;A(Z);: LET T=T
 +4: IF T>30 THEN  LET T=0
 60 NEXT Z
 70 PRINT : PRINT : PRINT "PRE
 SSIONE 'O' PARA ORDENAR"
 80 LET K$=INKEY$: IF K$<>"O"
 THEN  GOTO 80
 90 GOSUB 4000
100 PRINT : PRINT "TABELA ORDE
NADA": PRINT
110 LET T=0: FOR Z=1 TO AA
 120 PRINT TAB T;A(Z);: LET T=T
 +4: IF T>30 THEN  LET T=0
 130 NEXT Z
 140 GOTO 10
3999  REM ORDENACAO POR SUBSTIT
UICAO RETARDADA
4000 FOR I=1 TO AA-1
4010 LET K=I
4020 FOR J=I+1 TO AA
4030 IF A(J)<A(K) THEN  LET K=J
4040 NEXT J
4050 IF I<>K THEN  LET T=A(K):
LET A(K)=A(J): LET A(I)=T
4060 NEXT I
4070 RETURN
```



```
10 PRINT:PRINT:INPUT "NUMERO DE
 ITENS";AA:IF AA<2 THEN 10
15 DIM A(AA)
20 PRINT:PRINT"TABELA DESORDENA
DA":PRINT
30 FOR Z=1 TO AA
40 A(Z)=RND(100)
50 PRINT A(Z);
60 NEXT Z
70 PRINT:PRINT:PRINT "PRESSIONE
 'O' PARA ORDENAR"
80 K$=INKEY$:IF K$<>"O" THEN 80
90 GOSUB 4000
100 PRINT:PRINT:PRINT "TABELA O
RDENADA"
110 PRINT:FOR Z=1 TO AA
```

- ORDENAÇÃO POR SUBSTITUIÇÃO RETARDADA
- ORDENAÇÃO POR ESPALHAMENTO: FAZ O QUE SE FARIA MANUALMENTE
- RÁPIDA E SIMPLES: A ORDENAÇÃO DO JOGADOR DE CARTAS OU POR INSERÇÃO
- ORDENAÇÃO INSTANTÂNEA: A MAIS RÁPIDA DE TODAS

```
120 PRINT A(Z);
130 NEXT Z
140 RUN
3999 ORDENACAO POR SUBSTITUICAO
 RETARDADA
4000 FOR I=1 TO AA-1
4010 K=I
4020 FOR J=I+1 TO AA
4030 IF A(J)<A(K) THEN K=J
4040 NEXT J
4050 IF I<>K THEN T=A(K):A(K)=A
(I):A(I)=T
4060 NEXT I
4070 RETURN
```



Para que o programa acima rode no MSX substitua as seguintes linhas:

```
5 R=RND(-TIME)
40 A(Z)=INT(RND(1)*100+1)
50 PRINTA(Z),
120 PRINTA(Z),
4050 IF I<>K THEN SWAP A(K),A(I
)
```



Para que o programa acima rode nos micros da linha Apple II e TK-2000, substitua as linhas a seguir:

```
40 A(Z)=INT(RND(1)*100+1)
50 PRINTA(Z),
80 GET K$.IF K$<>"O" THEN 80
120 PRINTA(Z),
```

ORDENAÇÃO POR ESPALHAMENTO

A ordenação por espalhamento é outra rotina cuja velocidade se aproxima à da ordenação tipo bolha quando ambas são usadas para listas parcialmente ordenadas. Neste caso, uma matriz secundária é criada para uma ordenação preliminar parcial. Os valores inicial e final são determinados no início, enquanto outros itens só podem ser adicionados à lista quando esta estiver totalmente ordenada.

Embora essa rotina seja relativamente rápida, o uso de uma matriz secundária para armazenar dados diminui a memória disponível do micro.

Note que se deve especificar o valor máximo, o que é feito por meio da linha 5010. No nosso caso, esse valor é 100, que é o maior número aleatório permitido na linha 40.

Adicione as linhas seguintes ao primeiro programa:

S

```
  90 GOSUB 5000
4999 REM ORDENACAO POR ESPALHAM
ENTO
5000 DIM B(1.2*AA+30)
5010 FOR J=1 TO AA: LET K=INT (
A(J)*AA/100)+1
5020 IF B(K)=0 THEN  LET B(K)=A
(J): NEXT J· GOTO 5040
5030 LET K=K+1· GOTO 5020
5040 LET J=1· FOR K=1 TO 1 2*AA
+30: IF B(K)=0 THEN  NEXT K· GO
TO 5060
5050 LET A(J)=B(K)· LET J=J+1·
NEXT K
5060 FOR J=AA-1 TO 1 STEP -1: L
ET F=-1
5070 FOR K=1 TO J
5080 IF A(K)>A(K+1) THEN  LET F
=0: LET T=A(K): LET A(K)=A(K+1)
: LET A(K+1)=T
5090 NEXT K: IF F=0 THEN  NEXT
J: RETURN
```



```
90 GOSUB 5000
4999 REM **ORDENACAO POR ESPALH
AMENTO**
5000 DIM B(1.2*AA+30)
5010 FOR J=1 TO AA:K=INT(A(J)*A
A/100)+1
5020 IF B(K)=0 THEN B(K)=A(J):N
EXT:GOTO 5040
5030 K=K+1:GOTO 5020
5040 J=1:FOR K=1 TO 1.2*AA+30:I
F B(K)=0 THEN NEXT:GOTO 5060
5050 A(J)=B(K):J=J+1:NEXT
5060 FOR J=AA-2 TO 1 STEP -1:F=
-1
5070 FOR K=1 TO J+1
5080 IF A(K)>A(K+1) THEN F=0:T=
A(K)·A(K)=A(K+1):A(K+1)=T
5090 NEXT:IF F=0 THEN NEXT
5100 RETURN
```

Os usuários do MSX podem trocar a segunda metade da linha 5080, que faz a troca de valores entre as variáveis **A(K)** e **A(K+1)** para **SWAP A(K), A(K+1)**. O valor 1.2 na linha 5000 pode ser ajustado para fornecer o espaço necessário à matriz. Essa rotina de ordenação imita, de certa forma, a maneira normalmente usada para ordenar um grupo de informações: uma vez misturados os dados, e determinados o maior e o menor valor, tudo vai sendo posicionado conforme as prioridades.

ORDENAÇÃO POR INSERÇÃO

Uma opção para muitas aplicações é a rotina de ordenação por inserção. Supondo que os números abaixo sejam cartas de baralho, disponha-os na ordem dada, da esquerda para a direita:

```
esq.                dir.
 9    4    5    7    2
```

O processo começa a partir da esquerda (ou pelo primeiro valor da lista a ser ordenada). Ele procura pela primeira ocorrência de um número menor fora de ordem. Um rápido exame de linha mostra o 4 fora de lugar. Ele é então reposicionado (inserido) antes do 9, formando a nova sequência:

```
esq.                dir.
 4    9    5    7    2
```

As cartas 4 e 9 estão agora na sequência correta, mas, quando incluímos os outros números (5, 7 e 2) em nossa ob-

servação, a ordem desaparece. A cada passada, contudo, as cartas vão sendo reordenadas, até que alcancemos a sequência correta:

esq				dir
4	9	5	7	2
4	5	9	7	2
4	5	7	9	2
2	4	5	7	9

Como se pode ver, não há aqui separação de grupos ou comparações par a par de valores, como na maioria dos métodos já estudados: as cartas (números) são alinhadas da esquerda para a direita em ordem crescente, depois de algumas trocas de posição.

Esse processo é muito semelhante ao usado por um jogador de baralho ao analisar e ordenar uma mão de cartas. De fato, a ordenação por inserção é também conhecida como ordenação do jogador de cartas.

Vejamos agora um grupo de números um pouco maior, num estágio em que alguns valores já foram ordenados:

```
grupo não ordenado        g  ordenado
                               34
                               47
                               59
                               87
      102> > > > > > > >
       26                     144
       73                     167
      193
```

O número 102 é colocado na sua posição correta e a rotina prossegue deslocando os valores remanescentes:

```
grupo não ordenado        g. ordenado
       26> > > > > > > >
       73                      34
      193                      47
                               59
                               87
                              102
                              144
                              167
```

Em seguida, o número 73 é inserido na sua posição; resta agora, como se pode observar, apenas um valor não ordenado:

```
grupo não ordenado        g. ordenado
                               26
                               34
                               47
                               59
       73> > > > > > > >
      193                      87
                              102
                              144
                              167
```

E, para completar:

```
                               26
                               34
                               47
                               59
                               73
                               87
                              102
                              144
                              167
      193> > > > > > > >
```

Adicione as linhas abaixo ao programa de demonstração para observar a velocidade dessa rotina de ordenação:



```
  90 GOSUB 6000
5999 REM ORDENACAO POR INSERCAO
6000 FOR I=1 TO AA 1
6010 LET K=A(I+1)
6020 FOR J=I TO 1 STEP -1
6030 IF K>=A(J) THEN  GOTO 6070
6040 LET A(J+1)=A(J)
6050 NEXT J
6060 LET J=0
6070 LET A(J+1)=K
6080 NEXT I: RETURN
```



```
90 GOSUB 6000
5999 REM **ORDENACAO POR INSERC
AO**
6000 FOR I=1 TO AA 1
6010 K=A(I+1)
6020 FOR J=I TO 1 STEP -1
6030 IF K>=A(J) THEN 6070
6040 A(J+1)=A(J)
6050 NEXT J
6060 J=0
6070 A(J+1)=K
6080 NEXT I:RETURN
```

O valor da variável **K** é o próximo número da lista não ordenada a ser comparado. O laço externo que começa na linha 6000 "varre" a lista ordenada, passando dos menores para os maiores valores, até encontrar a posição para **K**. Então, a rotina das linhas 6020 a 6050 expande a lista ordenada para que haja lugar para o novo item. O programa continua até que todos os valores da lista não ordenada tenham sido colocados nos seus lugares.

ORDENAÇÃO INSTANTÂNEA

Embora seja considerada bastante rápida, a rotina de ordenação por inserção parece lenta quando comparada com a rotina de ordenação instantânea (ou *quicksort*, se você preferir). Esta última, contudo, além de ser complexa, exige muito espaço de memória. Assim, ela não é encontrada com grande frequência em programas especialmente desenhados para as limitadas memórias dos micros domésticos.

Seja como for, a ordenação instantânea é, sem dúvida, a mais veloz de todas as rotinas que já examinamos até agora. Além disso, ela vai mais longe do que uma simples rotina de comparação e troca. E, embora seu algoritmo seja bastante complexo (a ponto de não podermos explicá-lo aqui), vale a pena conhecê-la e usá-la!



```
  90 GOSUB 7000
6999 REM ORDENACAO INSTANTANEA
7000 LET K=0: LET I=0: DIM S(AA
)
7010 LET S(I+1)=1: LET S(I+2)=A
A
7020 LET K=K+1
7030 IF K=0 THEN  RETURN
7040 LET K=K-1: LET I=K+K
7050 LET A=S(I+1): LET B=S(I+2)
7060 LET Z=A(A): LET U=A: LET L
=B+1
7070 LET L=L-1
7080 IF L=U THEN  GOTO 7150
7090 IF Z<=A(L) THEN  GOTO 7070
7100 LET A(U)=A(L)
7110 LET U=U+1
7120 IF L=U THEN  GOTO 7150
7130 IF Z>=A(U) THEN  GOTO 7110
7140 LET A(L)=A(U): GOTO 7070
7150 LET A(U)=Z
7160 IF B-U>=2 THEN  LET I=K+K
 LET S(I+1)=U+1: LET S(I+2)=B:
LET K=K+1
7170 IF L-A>=2 THEN  LET I=K+K
 LET S(I+1)=A: LET S(I+2)=L-1:
LET K=K+1
7180 GOTO 7030
```



```
90 GOSUB 7000
6999 REM **ORDENACAO INSTANTANE
A**
7000 K=0:I=0:DIM S(AA)
7010 S(I+1)=1:S(I+2)=AA
7020 K=K+1
7030 IF K=0 THEN RETURN
7040 K=K-1:I=K+K
7050 A=S(I+1):B=S(I+2)
7060 Z=A(A):U=A:L=B+1
7070 L=L 1
7080 IF L=U THEN 7150
7090 IF Z<=A(L) THEN 7070
7100 A(U)=A(L)
7110 U=U+1
7120 IF L=U THEN 7150
7130 IF Z>=A(U) THEN 7110
7140 A(L)=A(U):GOTO 7070
7150 A(U)=Z
7160 IF B-U>=2 THEN I=K+K:S(I+1
)=U+1:S(I+2)=B:K=K+1
7170 IF L-A>=2 THEN I=K+K:S(I+1
)=A:S(I+2)=L-1:K=K+1
7180 GOTO 7030
```

PROGRAMAÇÃO DE MELODIAS NO MICRO

- CONVERSÃO DE PARTITURAS EM PROGRAMAS
- ACIDENTES
- RITMO
- EXECUÇÃO DE MELODIAS

Seu micro é capaz de executar, sozinho, melodias inteiras. Aprenda a converter partituras em linhas de programas BASIC e ouça, então, suas canções prediletas.

A música compõe-se de dois elementos básicos: som e ritmo. Nosso primeiro artigo sobre música em micros (veja página 721) limitou-se à produção de notas musicais. O computador emitia uma nota conforme a tecla pressionada e o ritmo ficava a cargo do usuário. Mostraremos agora como informar ao computador o ritmo desejado. Além disso, daremos as noções de notação musical necessárias à tradução de qualquer partitura, o que lhe permitirá executar no micro sua melodia predileta.

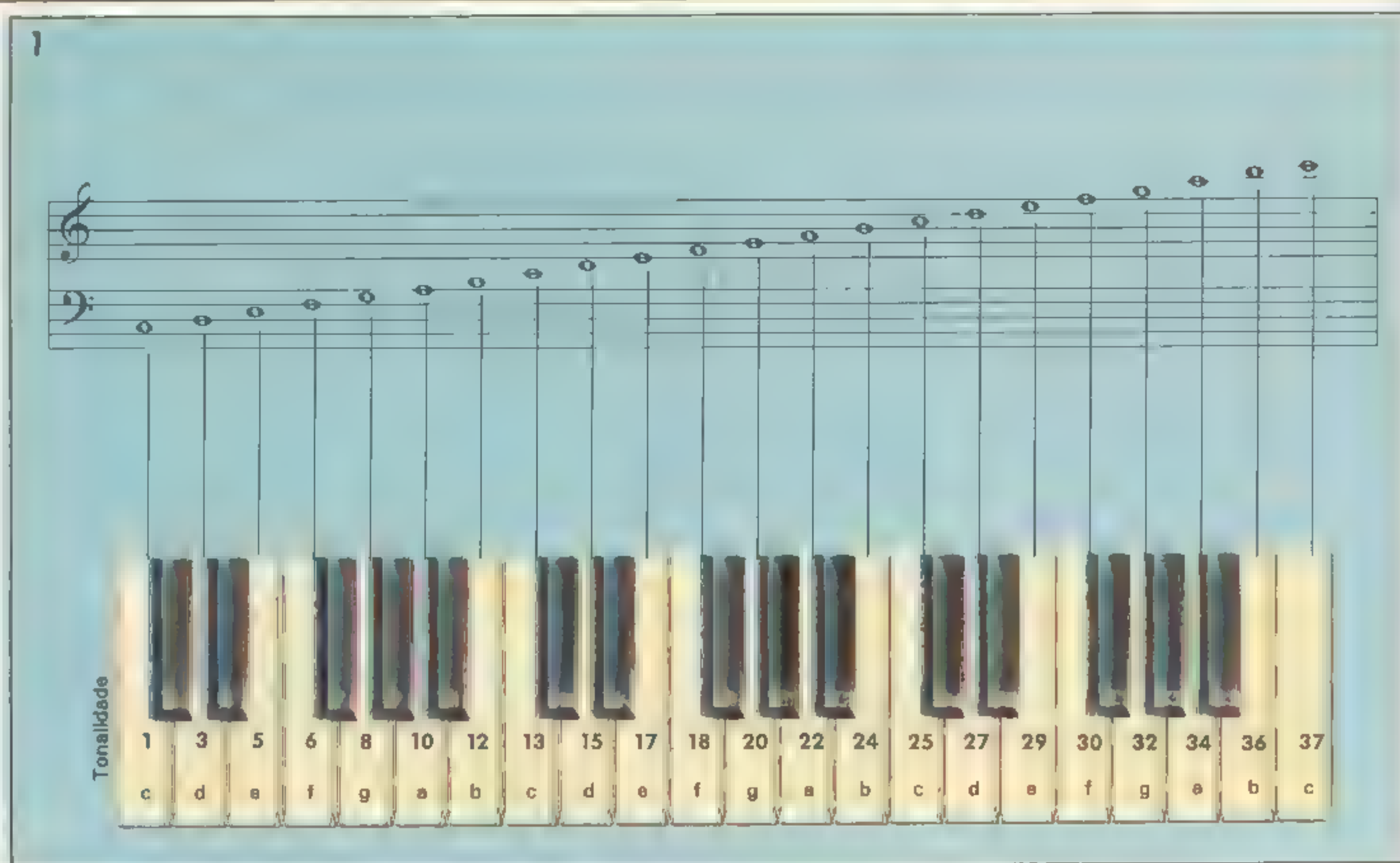
NOTAÇÃO MUSICAL

Em geral, as partituras musicais são escritas em um ou dois grupos de cinco linhas horizontais, paralelas e eqüidistantes, denominados *pauta*. Sobre uma ou entre duas linhas colocam-se os símbolos que representam as notas. A posição destas na pauta, no sentido vertical, determina a tonalidade; no sentido horizontal, a ordem de execução. O formato dos símbolos, por sua vez, determina a duração da nota.

Notas alinhadas verticalmente devem ser tocadas ao mesmo tempo, em acordes; estes, porém, não foram incluídos em nossos programas, que só tocam uma nota de cada vez.

Linhas verticais, as *barras* dividem a música em grupos de notas. A duração de cada um desses grupos deve ser a mesma, embora cada um deles possa ter um número diferente de notas.





TONALIDADE

O programa que apresentamos no artigo anterior transformou seu computador em uma espécie de instrumento musical. Como você já deve ter observado, ele funciona muito bem. Contudo, o número de melodias que pode executar é reduzido, em razão do pequeno número de notas disponíveis. Além disso, o usuário, que, muitas vezes, jamais tocou qualquer tipo de instrumento, é obrigado a executar cada música.

Pode-se facilmente converter uma partitura nos números que o computador precisa para tocá-la, mesmo sem ter grandes informações sobre música. Na página 743, fornecemos uma tabela de conversão justamente com esta finalidade. Antes, porém, devemos ampliar um pouco nossos conhecimentos a respeito de notação musical.

LER EM VEZ DE OUVIR

A figura 1 exibe um trecho de partitura. Nela estão representadas todas as notas disponíveis no teclado, em uma série crescente — é a chamada *escala cromática*.

Os símbolos no início das pautas são denominados *claves*: o da pauta de cima é a *clave de sol*, mais aguda; o da pauta de baixo, a *clave de fá*, mais grave. As claves determinam a tonalidade da pauta. Sem uma clave, as mesmas notas poderiam ser tocadas numa tonalidade mais alta ou mais baixa.

Os símbolos das notas ficam sempre entre duas ou exatamente sobre uma das linhas da pauta. Para escrever as notas que ficam acima ou abaixo destas, usam-se *linhas suplementares*. Na figura 1, por exemplo, a oitava e as três últimas notas estão desenhadas sobre linhas suplementares.

Para colocar as notas da partitura em um programa BASIC é necessário calcular os valores da duração bem como de tonalidade correspondentes a cada uma delas. Qualquer pessoa pode se confundir nessa demorada tarefa. Mas não desanime: há uma maneira mais fácil de realizá-la.

Voltemos à figura 1: observe que cada tecla foi marcada com um valor de 1 a 37. Esses números serão usados para calcular aqueles que, efetivamente, farão parte do programa.

Poderíamos consultar a mesma figura sempre que quiséssemos "traduzir" uma melodia, mas o processo de comparação entre a partitura e o desenho também seria bastante cansativo. Para simplificar o trabalho, o mais prático seria o uso de um dispositivo ou "gabarito" que, colocado sobre qualquer nota de partitura, mostrasse o valor correspondente à sua tonalidade.

UM GABARITO MUSICAL

Não há nenhum segredo na construção de um dispositivo desse tipo: precisamos simplesmente desenhar uma escala musical numa folha de papel, o que leva alguns poucos minutos. Provavelmente, melodias diferentes exigirão gabaritos diferentes, já que o tamanho das pautas pode variar.

A primeira coisa a fazer é pôr o papel sobre a partitura e copiar as linhas das pautas, indicando todas as possíveis posições verticais de uma nota. Em seguida, basta assinalar o valor correspondente a cada marca. Para identificar os valores que deveremos usar nos programas, colocamos o gabarito sobre a partitura e lemos nota por nota com sua ajuda.

Os valores correspondentes a cada tonalidade estão representados na tabela de conversão. O Spectrum, o Apple e o

TABELA DE CONVERSÃO

Escala principal	Spectrum	Apple	TK-2000	TRS-Color	MSX
1	12	192	192	C	C
2	−11	182	182	C+	C+
3	−10	172	172	D	D
4	−9	162	162	D+	D+
5	−8	154	154	E	E
6	−7	146	146	F	F
7	−6	137	137	F+	F+
8	−5	128	128	G	G
9	−4	121	121	G+	G+
10	−3	114	114	A	A
11	−2	108	108	A+	A+
12	−1	102	102	B	B
13	0	96	96	C	C
14	1	90	90	C+	C+
15	2	85	85	D	D
16	3	80	80	D+	D+
17	4	76	76	E	E
18	5	72	72	F	F
19	6	67	67	F+	F+
20	7	64	64	G	G
21	8	60	60	G+	G+
22	9	56	56	A	A
23	10	53	53	A+	A+
24	11	50	50	B	B
25	12	47	47	C	C
26	13	45	45	C+	C+
27	14	42	42	D	D
28	15	40	40	D+	D+
29	16	37	37	E	[illegible]
30	17	35	35	F	F
31	18	33	33	F+	F+
32	19	31	31	G	G
33	20	29	29	G+	G+
34	21	28	28	A	A
35	22	26	26	A+	A+
36	23	25	25	[illegible]	B
37	24	23	23	C	C

TK-2000 usam números, enquanto o MSX e o TRS-Color usam letras. Para saber o que escrever junto a cada marca do gabarito, compare os números da figura 1 com os da tabela e procure o valor correspondente na coluna dedicada ao seu micro.

BEMÓIS E SUSTENIDOS

Nem todas as melodias são escritas na escala "C maior" ("dó maior", como vimos no artigo anterior); portanto, às vezes precisaremos traduzir os bemóis e sustenidos das partituras.

Se observarmos os acidentes fixos — explicados mais adiante —, distinguiremos os bemóis ou sustenidos da canção. Poderemos, assim, substituir o valor da nota natural no gabarito pelo valor do bemol ou do sustenido.

Bemóis e sustenidos são as teclas pretas, e o valor de suas tonalidades está na tabela. O menor valor de um C sustenido é 2 (usando os números da escala principal da tabela, você descobrirá o valor correspondente em sua máquina). D bemol tem o mesmo valor, que está entre 1 e 3 (valores de C e D, respectivamente).

Símbolos para bemóis e sustenidos não foram introduzidos na figura 1 para não torná-la confusa. Quando se quer representar o sustenido de uma nota, ela aparece precedida pelo símbolo correspondente (parecido com o caracter # "cardinal" de seu computador). Da



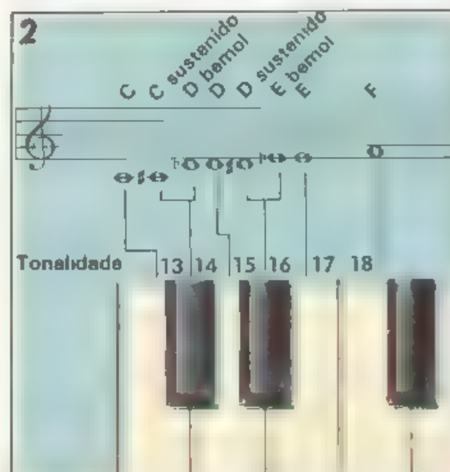
Como o computador produz sons?

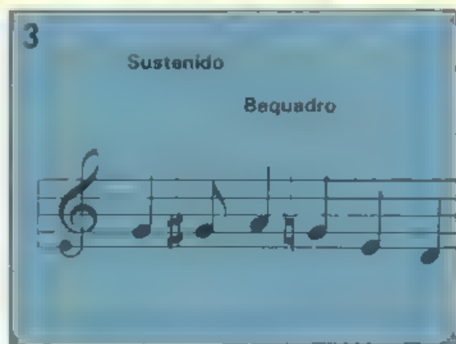
Para que um dispositivo digital, como o computador, possa gerar um fenômeno analógico (continuamente variável em intensidade), como o som é necessário ligá-lo a um aparelho especial, chamado *conversor digital-analógico* (DA). Este tem a função de converter uma lista de números armazenada ou gerada no computador em uma onda continuamente variável.

No nosso caso, cada número da lista representa a *amplitude* ou intensidade da onda em um determinado momento. A lista de números, com valores de amplitude tomados a intervalos regulares (geralmente milissegundos), constitui, assim, uma amostragem da onda a ser produzida. O inverso do intervalo de amostragem, denominado freqüência de amostragem, dependerá da freqüência máxima que o som deve atingir.

O conversor analógico-digital (AD) funciona de modo inverso ao conversor DA, transformando uma onda analógica em uma lista de números. Esse dispositivo é utilizado, por exemplo, em sistemas de reconhecimento de voz para microcomputadores ou na conversão de entradas provenientes de joysticks analógicos.

O conversor DA só pode ser empregado nos micros que dispõem de geradores internos de sons, como o TRS-Color, o MSX e o Spectrum. Colocando-se vários conversores DA lado a lado, a placa geradora de som desses micros torna-se capaz de emitir várias notas ao mesmo tempo.





mesma maneira, símbolos de bemol (parecidos com um b minúsculo) precedem as notas correspondentes.

A figura 2 mostra as primeiras notas "C maior", incluindo bemóis e sustenidos, na clave de sol. Observe que C sustenido e D bemol são a mesma nota, mas ocupam posições verticais diferentes na pauta. Notas isoladas podem, assim, representar bemóis e sustenidos, desde que sejam precedidas pelos símbolos correspondentes — denominados *acidentes*. O efeito de um acidente dura, para a nota que o segue, até a próxima barra, a menos que seja cancelado antes do fim do compasso. É um terceiro símbolo, chamado *bequadro*, que cancela o acidente e restaura a tonalidade natural da nota. A figura 3 mostra um trecho de partitura onde aparece um G sustenido. O G seguinte tem sua tonalidade natural preservada pelo bequadro que o precede.

ACIDENTES FIXOS

Quando uma canção é executada em uma escala que inclui um bemol ou um sustenido, essa nota será usada o tempo todo, no lugar do valor natural correspondente. Para evitar a repetição do símbolo, ele é colocado após a clave, no início da pauta. Nesse local, o símbolo é chamado de *acidente fixo* e, por meio dele, pode-se identificar a escala em que é tocada a canção.

A ausência de um acidente fixo indica que a escala é "C maior", e, ao contrário, qualquer acidente fixo indica que se trata de outra escala. Se, por exemplo, uma pauta apresenta um sustenido como acidente fixo na linha do F, todos os F, em todas as oitavas, serão sustenidos.

No artigo anterior, mostramos que um F sustenido é necessário para preservar o arranjo de intervalos na escala "B maior". Um sustenido na posição do F como acidente fixo indica ao músico que a melodia foi escrita em "B maior". Não se esqueça, portanto, de modificar no gabarito os valores das tonalidades correspondentes, sempre que a partitura tiver acidentes fixos.

A figura 4 mostra um trecho de partitura em "F maior". Nele vemos um acidente fixo bemol. Os valores das tonalidades estão indicados: note que o da quarta nota, um B bemol, é 23. Aqui o gabarito revela toda a sua utilidade. Se a música que está sendo transcrita tem um B bemol como acidente fixo, colocamos o valor correspondente ao bemol na marca do B. Assim, a tradução dos acidentes se torna tão fácil quanto a das notas naturais.

Mas tenha cuidado: muitas vezes alguns acidentes isolados — e naturais isolados — têm prioridade sobre os acidentes fixos.

RITMO

Depois de calcular a tonalidade das notas, devemos cuidar da duração de cada uma delas e da tradução das pausas, que têm seus próprios símbolos. A figura 5 mostra os símbolos das notas e das pausas com duração equivalente e o valor relativo da duração.

Durações relativas são utilizadas para que um *andamento* — velocidade de execução — defina, posteriormente, a verdadeira duração. Podemos, assim, experimentar diversos andamentos, até que um deles nos soe melhor. O programa do final do artigo permitirá esse tipo de ajuste.

As pausas mais longas são representadas por meio de pequenos retângulos pretos, desenhados acima ou abaixo da linha média da pauta. Quando um ponto sucede uma nota ou uma pausa, sua duração é multiplicada por um e meio. Uma nota pontuada duas vezes tem uma duração uma vez e três quartos maior que a nota sem pontos.

Além do acidente fixo, junto à clave aparecem dois algarismos, dispostos como fração numérica. O de baixo indica a espécie de notas em que o compasso está dividido; o de cima, o número dessas notas no compasso. Na figura 6, por exemplo, vemos a fração 6/8 depois da clave, o que significa que há seis colcheias por compasso.

Para nossos fins essas noções de notação musical são suficientes: vamos apenas transcrever a partitura para criar um programa.

TRADUÇÃO DE UMA MELODIA

Vamos agora traduzir uma pequena partitura em números (ou letras, conforme sua máquina) que o computador possa utilizar para determinar tonalidades e durações de notas.

A figura 6 mostra a partitura de uma canção infantil inglesa, *Three Blind Mice* (*Três Ratinhos Cegos*) acompanhada dos números que representam a tonalidade na escala principal, previamente definida. Para identificar os números correspondentes em seu computador, use a tabela de conversão. Os valores para a duração relativa também estão indicados.

Observando a partitura em questão, você verá que, sempre que três colcheias (veja figura 5) aparecem juntas, elas são ligadas por um traço horizontal. Não estranhe: a única finalidade dessa forma de notação — não obrigatória — é facilitar a leitura. As notas permanecem exatamente as mesmas, não havendo modificações nem na altura nem na duração.

Já no oitavo compasso, há duas notas ligadas, mas por uma linha curva, e não reta. Nesse caso há mudança: a linha, chamada *ligadura*, indica que a segunda nota não deve soar separadamen-

MICRO DICAS

SELEÇÃO DE PARTITURAS

Se você sabe ler partituras, não terá a menor dificuldade em encontrar as músicas de sua preferência para "tocar" no computador, usando o programa deste artigo. Mas, se você dispõe apenas de noções elementares de notação musical, precisará contornar suas deficiências selecionando partituras simplificadas. Uma excelente fonte são os álbuns de exercícios musicais para iniciantes de instrumentos menos complexos, como a flauta doce. Neles, melodias não harmonizadas (isto é, sem acordes) são apresentadas na forma de partituras tão simples que você não terá dificuldade nenhuma em acompanhar.

A seleção de músicas será um pouco mais difícil para quem não lê absolutamente nada de uma partitura. Mas, ainda assim resta o popular recurso de "tocar por números". Alguns fabricantes de instrumentos musicais para crianças publicam partituras em que as notas são acompanhadas por números. Estes indicam a tecla a ser pressionada. Estabelecendo a correspondência entre os números e as teclas do computador, você poderá executar músicas completas sem saber sequer o que é um "dó".

4

Acidente fixo



Tonalidade 18 22 20 23 22 25 22 18

te, ou seja, que o som da primeira se prolongará pelo tempo correspondente à duração de ambas as notas. Poderíamos ter, também, ligaduras com várias notas. Para transcrevê-las em dados computáveis, bastaria somar as durações correspondentes, o que nos daria uma nota mais longa.

LINHAS DATA

Os números das tonalidades e durações devem ser colocados em linhas **DATA**, para que o programa possa obtê-los com **READ** no decorrer da execução. Os números correspondentes à canção da figura 6 foram incorporados ao programa dessa maneira. Cada linha contém dois compassos (exceto no TRS-Color e no MSX), o que torna mais fácil a compreensão das linhas **DATA**.

A duração total de cada compasso deve ser a mesma — doze unidades, no nosso caso. Assim, se você transcrever uma música e o ritmo parecer incorreto, verifique se cada compasso tem a mesma duração, corrigindo os erros encontrados. Use as funções de edição de seu computador para duplicar compassos iguais ou semelhantes, economizando tempo de digitação.

ANDAMENTO

Os valores da duração de cada nota são relativos, já que podemos variar o andamento (velocidade) que se imprime à execução de um trecho da melodia. O programa solicita um valor para o andamento, logo que é executado. Como valores menores correspondem a maiores velocidades de execução, podemos considerar o valor que fornecemos como o inverso do andamento.

Há um detalhe importante sobre o funcionamento do programa que se aplica a todos os micros. Suponhamos que a melodia contenha um compasso com uma única nota bem longa, seguido por outro compasso com várias notas curtas. No segundo caso, as linhas que lêem e testam os dados são executadas mais vezes que no primeiro. Devido ao tempo de operação extra, a velocidade de execução do segundo compasso será um pouco menor. Deve-se imprimir, portanto, a maior velocidade possível à execução dessas linhas BASIC. Por isso, não há no programa linhas que testem o fim dos dados: o programa termina simplesmente com uma mensagem de erro indicando que não há mais dados.

Se, porém, ao traduzir suas músicas prediletas, você quiser um final mais elegante, coloque um laço **FOR...NEXT** que leia exatamente a quantidade de números que as linhas **DATA** contêm.

Seguem-se os programas que executam a partitura da figura 6.

```
10 INPUT "TEMPO (1-50)",t
LET t=t/100
20 IF t<0 001 OR t>0 5 THEN
GO TO 10
30 FOR n=0 TO 1 STEP 0  READ
a,b  SOUND a*t,b  NEXT n
100 DATA 6,4,6,2,12,0,6,4,6,2,
12,0
110 DATA 6,7,4,5,2,5,12,4,6,7,
4,5,2,5,10,4,2,7
120 DATA 4,12,2,12,2,11,2,9,2,
11
130 DATA 4,12,2,7,4,7,2,7
140 DATA 2,12,2,12,2,12,2,11,2
,9,2,11
150 DATA 4,12,2,7,4,7,2,7
160 DATA 2,12,2,12,2,12,2,11,2
,9,2,11
170 DATA 4,12,2,7,4,7,2,5
180 DATA 6,4,6,2,12,0
```

Ao rodar o programa, devemos responder, com um número entre 1 e 50, à solicitação do computador de um valor para o andamento. Como já foi explicado, trata-se do inverso do andamento. O programa nada mais faz do que multiplicar cada duração pelo valor do andamento — portanto, quanto maior o valor, menor a velocidade.

Usamos o inverso do andamento para tornar o programa mais simples e rápido. Um comando **IF...THEN** verifica se o andamento está na faixa permitida. A linha 30 executa a melodia. Um laço **FOR...NEXT** obtém com **READ**

5

Símbolo da nota	Nome da nota	Símbolo de pausa	Duração relativa	Duração quando pontuada
	Semibreve		16	24
	Mínima		8	12
	Semínima		4	6
	Colcheia		2	3
	Semicolcheia		1	1 5

6

Tonalidade	17	15	13	17	15	13	20	18	18	17
Duração	6	6	12	6	6	12	6	4	2	12

20 18 18 17 20 25 25 24 22 24 25 20 20 20
6 4 2 10 2 4 2 2 2 2 4 2 4 2

25 25 25 24 22 24 25 20 20 20 20 25 25 24 22 24
2 2 2 2 2 2 4 2 2 2 2 4 2 2 2 2

25 20 20 20 18 17 15 13
2 2 2 4 2 6 6 12

dois valores para cada nota: um para a tonalidade e outro para a duração. Um comando **SOUND** emite o som antes que outros dados sejam lidos.

Ao executar suas próprias canções, acerte os valores do laço **FOR...NEXT** para a leitura do número exato de notas. Por enquanto, o laço nunca termina, pois tem um **STEP** de valor 0.



```
10 INPUT "ANDAMENTO (1 50)";TP
20 IF TP<1 THEN 10
30 PLAY "T"+STR$(INT(32+223/TP)
)
40 FOR I=1 TO 48
50 READ A$,D
60 PLAY "L"+STR$(INT(1+63/D))+"
03"+A$
70 NEXT I
1000 DATA E,6,D,6,C,12
1010 DATA E,6,D,6,C,12
1020 DATA G,6,F,4,F,2,E,12
1030 DATA G,6,F,4,F,2,E,10
1040 DATA G,2,04C,4,04C,2,B,2,A
,2,B,2,04C,4,G,2,G,4,G,2
1050 DATA 04C,2,04C,2,04C,2,B,2
,A,2,B,2,04C,4,G,2,G,2,G,2,G,2
1060 DATA 04C,4,04C,2,B,2,A,2,B
,2,04C,2,G,2,G,2,G,4,F,2
1070 DATA E,6,D,6,C,12
```

O programa do MSX começa pedindo que o usuário informe o andamento desejado. A linha 20 verifica se o valor escolhido é válido — se não for, o programa volta à linha 10. A linha 30 cuida de estabelecer o andamento, selecionando um valor adequado para o argumento "T" do comando **PLAY**. O valor escolhido no início deve ser transformado em cordão pela instrução **STR$**, para que possa servir como argumento de **PLAY**.

A linha 50 obtém os valores de tonalidade e duração das notas. A linha 60 produz o som, também usando o comando **PLAY**. Para determinar a duração de cada nota, **STR$** transforma em cordão o valor obtido nas linhas **DATA** e o coloca logo após "L". O "03" que se segue determina a oitava (eventualmente, é cancelado pelo "04" de uma nota). Finalmente, a variável A$ estabelece a altura da nota.

Para executar a musica, usamos um laço **FOR...NEXT** (linhas 40 a 70). Evitamos, assim, a emissão de uma mensagem de erro que, além de deselegante, causa problemas no MSX. Como esse computador possui um microprocessador específico para a produção de sons, a música continua tocando, mesmo após a execução do programa. Havendo uma mensagem de erro, o sistema operacional do MSX suspende imediatamente todas as atividades periféricas em andamento — tela gráfica, motor do gravador e som. Assim, se deixássemos passar um erro, a música seria interrompida.



```
10  HOME
20  FOR I = 0 TO 22: READ A: PO
KE 800 + I,A: NEXT
30  DATA  160,0,174,133,3,238,4
8,192,136,208,5,206,132,3,240,6
,202,208,245,76,34,3,96
40  INPUT "ANDAMENTO (1-50)",T
50 T = T / 3.3
60  READ A,B
70  POKE 900,1 + B * T: POKE 90
1,A: CALL 800
80  FOR I = 1 TO 100: NEXT
90  GOTO 60
100  DATA  76,6,85,6,96,12
110  DATA  76,6,85,6,96,12
120  DATA  64,6,72,4,72,2,76,12
130  DATA  64,6,72,4,72,2,76,10
,64,2
140  DATA  47,4,47,2,50,2,56,2,
50,2
150  DATA  47,4,64,2,64,4,64,2
160  DATA  47,2,47,2,47,2,50,2,
50,2,50,2
```



```
170  DATA  47,4,64,2,64,2,64,2,
64,2
180  DATA  47,4,47,2,50,2,56,2
,50,2
190  DATA  47,2,64,2,64,2,64,4,
72,2
200  DATA  76,6,85,6,96,12
```

O programa do Apple usa a rotina em código explicada no artigo da página 706. As linhas 20 e 30 colocam essa rotina na memória, depois que o **HOME** da linha 10 limpou a tela.

A linha 40 solicita o valor desejado para o andamento da canção e a linha 50 corrige seu valor. O fator de correção aqui escolhido é compatível com durações relativas com valores até 16 (semibreve). Como em nossa canção não existem notas com duração tão longa, podemos alterar o valor para 2.5, se quisermos. Nesse caso, porém, a duração máxima de cada nota deverá ser 12 (mínima pontuada).

A linha 60 obtém os valores de tonalidade e duração e à linha 70 cabe executar a nota. Para chamarmos a rotina (**CALL 800**) que produz o som, o endereço 900 deve conter a duração e o 901, a tonalidade.

O problema de tempo, que mencionamos ao tratar dos demais computadores, não se aplica ao micro Apple. A linha 80 produz um pequeno atraso, fazendo com que a música não seja executada muito rapidamente.

A linha 90 manda o programa de volta à linha 60 para que o programa obtenha o valor de uma nova nota. O programa termina com uma mensagem de erro, informando o fim dos dados.



Embora o programa anterior funcione no TK-2000, seus usuários devem fazer algumas modificações. Primeiro, como a rotina de produção de sons não é necessária, as linhas 20 e 30 podem ser apagadas. Além disso, devemos substituir os comandos na linha 70 do programa do Apple pelo comando **SOUND**. A linha 80 também precisa ser modificada, já que o TK-2000 é mais lento que o Apple. Aqui estão as modificações:

```
70 SOUND A, 1+B*T
80 FOR I=1 TO 40 NEXT
```

T

```
10 INPUT"ANDAMENTO (1-50)";TP
20 IF TP<1 THEN 10
30 READ A$,D
40 PLAY"T"+STR$(INT(1+255/(TP*D
)))+"O3"+A$
50 GOTO 30
1000 DATA E,6,D,6,C,12
1010 DATA E,6,D,6,C,12
1020 DATA G,6,F,4,F,2,E,12
1030 DATA G,6,F,4,F,2,E,12
1040 DATA O4C,4,O4C,2,B,2,A,2,B
,2,O4C,4,G,2,G,4,G,2
1050 DATA O4C,2,O4C,2,O4C,2,B,2
,A,2,B,2,O4C,2,G,2,G,2,G,2,G,2
1060 DATA O4C,4,O4C,2,B,2,A,2,B
,2,O4C,2,G,2,G,2,G,4,F,2
1070 DATA E,12,D,12,C,12
```

O programa do TRS-Color começa pedindo o valor da velocidade de execução desejada — ou andamento — por meio de um comando **INPUT**.

A linha 20 verifica se o valor fornecido está dentro dos limites válidos, retornando à linha 10 se o andamento for menor que 1.

A linha 30 lê dois valores para cada nota: um para a tonalidade — representada por meio de uma letra — e outro para a duração. Para simbolizar os sustenidos, usamos um sinal "cardinal" ou um "mais" antes da nota e, para os bemóis, um sinal "menos" no mesmo lugar. A linha 40 toca a nota, através de um comando **PLAY**. O **T** entre aspas faz com que o número que vem a seguir, entre 1 e 255, determine o andamento da melodia. Esse número é calculado invertendo-se o valor inicialmente fornecido ao programa. Para ser utilizado como argumento de **PLAY**, o número deve ser convertido em um cordão por intermédio do comando **STR$**.

O cordão "O3" seleciona a oitava apropriada; a parte final da linha (**+A$**) cuida da tonalidade da nota.

Após produzir o som de uma nota, o programa volta à linha 30, onde **READ** lê um novo par de dados nas linhas **DATA**. Isto se repete até que o conteúdo das linhas **DATA** se esgote. O programa é interrompido com uma mensagem de erro, ao tentar prosseguir a leitura de dados — que já acabaram.

UM VIDEOGAME EM ASSEMBLER

Começamos aqui a programação de um videogame completo: *Avalanche*. Em cada nova seção você encontrará uma surpresa... e aprenderá um pouquinho mais sobre código de máquina!

Os princípios mais importantes da linguagem de máquina podem ser ensinados por meio da programação de jogos. E, sem dúvida, aprender desse jeito é muito mais divertido. Desenvolvemos, assim, um videogame completo, especialmente construído para demonstrar os recursos de programação do Spectrum, do MSX e do TRS-Color. Não haverá programas para as linhas Apple, TK 2000, TRS-80 e ZX-81.

Avalanche é um videogame no estilo de *Keystone Kappers*, exigindo rapidez e agilidade do jogador, que precisa correr e saltar para escapar dos perigos que o ameaçam.

Serão criadas quatro telas, uma para cada nível de dificuldade — o jogo vai se tornando cada vez mais difícil. O personagem central de toda a ação é Willie. Nosso herói decide fazer um piquenique em uma montanha à beira-mar. Depois de encontrar um bom lugar para estender sua toalha, vai dar um passeio pelas redondezas. Ao voltar, descobre que alguns cabritos monteses espalharam seu lanche por toda a encosta. Para recuperar o que trouxe, precisará ir até o topo da montanha — o que deve fazer o mais rápido possível, levando-se em conta que a maré está subindo. Willie morrerá afogado se não chegar lá em cima a tempo.

Na primeira etapa do jogo, nosso herói é bombardeado por uma avalanche de pedras que rolam montanha abaixo. Para isso, utilizamos a primeira tela, que mostra as pedras descendo a encosta. Willie precisa saltá-las, se quiser continuar vivo — qualquer descuido pode ser fatal. A corrida e os saltos do personagem são controlados pelas teclas N e M. Se uma pedra atingi-lo, é morte certa. Felizmente, ele tem cinco vidas.

Quando Willie chega ao topo da montanha e recupera parte do que perdeu, é obrigado a voltar ao nível do mar. O jogo recomeça, então, na segunda tela. Desta vez, ao tentar subir o monte, encontrará pelo caminho enormes buracos, verdadeiras armadilhas cavadas na encosta. Se cair num deles ficará enterrado... e morto.

Chegando ao cume da segunda montanha, nosso herói será colocado aos pés da terceira e precisará enfrentar uma nova escalada. A encosta continua cheia de buracos, mas, agora, cada um deles é habitado por uma serpente venenosa, que tentará picar o personagem quando este saltar.

Na quarta tela, Willie terá que mostrar toda a sua habilidade, pois será bombardeado pela avalanche enquanto salta sobre os buracos habitados pelas serpentes "assassinas".

Nas quatro etapas do jogo a rapidez





- ESTRUTURA GERAL DO JOGO
- PROGRAMAS BASIC QUE AUXILIAM A MONTAGEM
- TABELAS DE DADOS EM ASCII
- CRIAÇÃO DE TELAS
- ROTINAS DA ROM
- IMPRESSÃO DE CARACTERES NA TELA
- PROGRAMAÇÃO DE PAUSAS
- CORREÇÃO DE ERROS



é fundamental, pois a maré está sempre subindo. Os pontos do jogador são contados conforme seu sucesso em escalar a montanha, havendo um bônus para quem conseguir passar as quatro telas sem perder nenhuma vida.

Além de ensinar os truques da programação em código de máquina, *Avalanche* é um jogo envolvente e cheio de emoções. Publicaremos suas seções em várias partes. O funcionamento de cada uma e seu papel na estrutura global do programa serão explicados detalhadamente na medida em que forem apresentadas. Sempre que possível, indicaremos as alternativas de utilização de determinadas rotinas. No final da série, você terá um videogame tão bom como os disponíveis em cartuchos.

A ESTRUTURA DO PROGRAMA

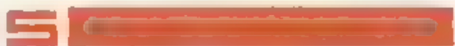
O cenário e os caracteres móveis são todos desenhados com blocos gráficos. Colocamos o cenário na tela por meio de laços que obtêm os caracteres correspondentes em bancos de blocos criados previamente na memória.

Os buracos e as cobras são superpostos ao cenário original, de maneira que não precisamos redesenhá-lo para montar as demais telas.

A parte principal do programa consiste em uma rotina de controle que determina o andamento e a prioridade dos eventos. Cada evento em si é executado por uma sub-rotina específica.

A animação do personagem central e das serpentes "assassinas" é feita em saltos de meio caractere, procedimento viabilizado pela criação de dois conjuntos de caracteres.

Com esse tipo de animação, pode-se obter suavidade nos movimentos sem complicar demais o programa ou atrasá-lo — o que comprometeria o desenvolvimento do jogo, já que a velocidade, no caso, é muito importante.



A primeira coisa que se programa em qualquer jogo é uma tela exibindo o título. Embora a rotina de impressão esteja em código de máquina, será mais fácil colocar as letras na memória usando um programa BASIC.

Antes de executar o programa, proteja a região da memória correspondente com **CLEAR 57434**.

```
10 LET X=57435
20 READ A$
30 FOR N=1 TO 38
40 POKE X,CODE A$(N)
50 LET X=X+1
60 NEXT N
70 DATA "AVALANCHECRIACAO A D
OEPROGRAMA:P.CLARK"
```

O programa cria na memória uma tabela com os códigos ASCII das letras que compõem a tela. Essa parte da memória deve ser gravada juntamente com o resto do programa em código; assim, na realidade, o programa BASIC não será necessário à execução do jogo. Grave-o, contudo, para auxiliar na montagem do programa completo.

Para que o Assembler de INPUT funcione a contento, faça as seguintes modificações: na linha 5120, troque o número 6 por 22; na linha 6110, adicione **:GOSUB 6150**; na linha 6150 troque **GOSUB 6260** por **GOSUB 6160**. Carregue, então, o Assembler e digite esta primeira rotina:

```
 40 REM org 58035
 50 REM ti call cl
 60 REM ld a,2
 70 REM out 254,a
 80 REM ld a,16
 90 REM ld (23624),a
100 REM ld ix,57435
110 REM ld b,9
120 REM ld a,70
130 REM ld hl,299
140 REM call me
150 REM ld b,15
160 REM ld a,7
170 REM ld hl,616
180 REM call me
190 REM ld b,18
200 REM ld hl,679
210 REM call me
220 REM ld b,2
230 REM ldp ld hl,65000
240 REM ld de,0
250 REM ldq dec hl
260 REM push hl
270 REM sbc hl,de
280 REM pop hl
290 REM jr nz,ldq
300 REM djnz ldp
301 REM nop
302 REM nop
303 REM nop
304 REM nop
305 REM nop
306 REM nop
307 REM nop
308 REM nop
310 REM ret
```

Grave os mnemônicos usando a opção correspondente do Assembler. Depois, monte a primeira rotina e grave o programa em código — se preciso, recorra ao monitor de linguagem de máquina. Digite, então, mais esta rotina:

```
 10 REM org 58146
 20 REM ktt ld a,253
 30 REM in a,254
 40 REM bit 1,a
 50 REM jr nz,ktt
 60 REM ret
 70 REM me push bc
 80 REM push af
 90 REM ld a,(ix+0)
100 REM call asc
110 REM pop af
120 REM call print
130 REM inc hl
140 REM inc ix
150 REM pop bc
160 REM djnz me
170 REM ret
180 REM asc push hl
190 REM ld hl,15608
200 REM ld de,8
210 REM ld b,31
220 REM sub b
230 REM ash add hl,de
240 REM dec a
250 REM jr nz,ash
260 REM push hl
270 REM pop bc
280 REM pop hl
290 REM ret
300 REM cl ld ix,16384
310 REM ld hl,6912
320 REM ld a,0
330 REM clp ld (ix+0),a
340 REM inc ix
350 REM dec hl
360 REM push hl
370 REM ld de,0
380 REM sbc hl,de
390 REM pop hl
400 REM jr nz,clp
410 REM ret
420 REM print push af
430 REM push hl
440 REM push bc
450 REM push hl
460 REM pop de
470 REM ld a,d
480 REM cp 1
490 REM jr c,next
500 REM push de
510 REM ld de,1792
520 REM add hl,de
530 REM pop de
540 REM ld a,d
550 REM cp 1
560 REM jr z,next
570 REM push de
580 REM ld de,1792
590 REM add hl,de
600 REM pop de
610 REM next push de
620 REM ld de,16384
630 REM add hl,de
640 REM pop de
650 REM ld a,8
660 REM pop bc
670 REM push af
680 REM rept ld a,(bc)
690 REM ld (hl),a
700 REM inc h
710 REM inc bc
720 REM pop af
730 REM dec a
740 REM jr z,ex.1
```

```
    REM push af
    REM jr rept
 70 REM exit pop hl
    REM pop af
    REM push de
    REM ld de,22528
    REM add hl,de
820 REM pop de
    REM ld (hl),a
    REM push de
    REM pop hl
    REM ret
```

Grave essas duas rotinas separadamente. O programa pode ser executado pelo comando **RAND USR 58035**, como de costume. Lembre-se, porém, de que a tabela com dados em ASCII — que começa em 57435 — deve estar na memória para que o programa funcione.



O PROGRAMA BASIC

O programa escrito em BASIC resume-se a um laço **FOR...NEXT** que coloca as letras das palavras da página inicial na memória, formando uma tabela ASCII que o programa em código pode utilizar. Ele apenas ajuda a montar o programa, não sendo necessário ao funcionamento do jogo depois que a tabela tiver sido gravada.

OS CÓDIGOS

O programa consiste em uma rotina principal que chama as demais rotinas. Podemos, assim, cuidar de cada rotina separadamente, o que facilita muito a detecção e correção de erros.

A primeira instrução **call cl** chama a sub-rotina de limpeza da tela. Os comandos **ld a,2** e **out 254,a** definem as cores da borda, como foi explicado na página 556. O uso do comando **out** modifica apenas temporariamente a cor da moldura. Para tornarmos essa alteração permanente, precisamos modificar o valor da variável do sistema correspondente, que fica na posição 23624.

A cor da moldura especificada no comando **out** é 2, ou vermelho. Mas, para que o 2 seja colocado em 23624, devemos "deslocá-lo" — operação **SHIFT** — três bits para a esquerda, resultando no valor 16.

IMPRESSÃO NA TELA

A rotina rotulada com **me** controla a impressão de caracteres na tela. Certos parâmetros devem ser fornecidos a ela para que possa imprimir o texto correto na posição desejada.

O comando **ld ix,57435** coloca no registro IX o endereço do primeiro byte da tabela ASCII, a fim de que a rotina de impressão saiba o que imprimir.

O acumulador contém o atributo do caractere que será impresso. Tanto em linguagem de máquina quanto em BASIC o funcionamento é o mesmo. Quando o bit 7 está ligado, o caractere é cintilante — **FLASH**. O bit 6 dá brilho ao caractere — **BRIGHT**. Os próximos três bits determinam a cor do fundo, e os três seguintes são responsáveis pela cor dos caracteres.

No programa-fonte, colocamos 70 — 01000110 em binário —, que tem o bit 6 ligado, os bits que controlam a cor do fundo nulos e os bits dos caracteres valendo 6. Teremos, assim, caracteres amarelos sobre fundo preto, brilhante e não cintilante.

O registro B funciona como contador de caracteres. O valor nele colocado determina o comprimento do texto que será impresso. Na primeira vez que a rotina **me** é chamada, B contém 9 — as nove letras de *AVALANCHE*.

O registro HL contém a posição da tela a partir da qual o texto deve ser impresso. Essa posição é calculada em números de posições a partir do canto esquerdo da tela. Quando HL contém 139, o primeiro caractere da primeira cadeia de caracteres — ou seja, o "A" de *AVALANCHE* — é colocado na quinta linha da tela, onze posições a partir do lado esquerdo.

A rotina **me** é chamada três vezes para imprimir cada uma das três linhas da tela inicial.

PROGRAMAÇÃO DE PAUSAS

Para que o jogador possa ler o título, devemos incluir uma pausa no programa. Colocamos 2 no registro B, de modo que o laço responsável pela pausa seja executado duas vezes.

O número 65000 é colocado no registro HL. O conteúdo desse par diminui em uma unidade a cada volta do laço interno — rotulado com **ldq.** HL é guardado e recuperado da pilha somente para ganharmos tempo.

Pode parecer estranho subtrair zero — conteúdo de DE — de HL a cada volta do laço. Mas trata-se apenas de um recurso utilizado para afetar o indicador de zeros, que não reage à instrução **pop.** O comando **jr nz** depende desse indicador para que o programa possa sair do laço. Quando HL for reduzido a zero e **sub hl, de** executar a próxima subtração, o resultado obtido será zero; ativado o indicador de zeros, o processador sairá do laço.

Quando completo, o programa pros-



seguirá imprimindo a página de instruções. Mas, por enquanto, a instrução **ret** provoca o retorno ao BASIC.

Não se importe com os comandos **nop** que aparecem no final do programa. Eles não têm nenhum efeito sobre o microprocessador e sua função aqui é somente preencher espaço.

ROTINAS AUXILIARES

Algumas rotinas de apoio ao programa principal foram montadas. A que tem o rótulo **ktt** aguarda que pressionemos uma certa tecla, e ainda não será usada. O princípio de verificação do teclado aqui utilizado é igual ao explicado no artigo da página 381.

A rotina **me**, encarregada de controlar a impressão na tela, começa guardando os conteúdos de BC e AF na pilha. Em seguida, o conteúdo da posição de memória apontada pelo registro IX (que contém o endereço inicial da tabela ASCII) é colocado no acumulador por **ld a,(ix + 0)**, comando que emprega o endereçamento indexado.

Os registros IX e IY são chamados de registros-índices, e dispõem de dezesseis bits para acomodar um endereço de memória. A sintaxe desse tipo de comando — muito utilizado, aliás, quando trabalhamos com tabelas — é **ix + d** ou **iy + d**, onde **d** é um valor entre 0 e 256 que pode ser somado ao endereço apontado pelo registro-índice.

Depois de colocar em A o código do caractere que deve ser impresso na tela, o programa chama a sub-rotina **asc**, que calcula a posição do padrão do caractere dentro de uma tabela de padrões existente na ROM (veja o artigo da página 381). O endereço inicial do padrão do caractere está em BC, quando o processador retorna da sub-rotina. Recupera-se o valor de AF da pilha e uma nova sub-rotina — **lprint**, que cuida da impressão propriamente dita — é chamada. Quando o processador retorna de **lprint**, HL contém a posição da tela onde houve impressão — na realidade, a posição do byte superior do padrão do caractere. A instrução **inc hl** aumenta este valor em uma unidade. O índice IX também é aumentado, passando a apontar para a próxima letra da tabela ASCII.

O par BC (B continha originalmente o comprimento do texto a ser impresso) é recuperado da pilha; enquanto seu conteúdo não for zero, a instrução **djnz me** diminui B em uma unidade e repete o processo de impressão.

A rotina **asc** calcula a posição do padrão do caractere a ser impresso. Em primeiro lugar, ela coloca temporariamente na pilha, por **push hl**, a posição de impressão que está em HL, pois vamos usar esse par de registros. Nele é colocado o endereço inicial da tabela de padrões — 15608. O número 8 é, então, colocado em DE e o 31, em B. A instrução **sub b** subtrai 31 de A. A operação é necessária porque a tabela de padrões começa com o caractere de código 32 — espaço em branco. Os códigos com valores menores que 32 são comandos BASIC e não caracteres.

Um caractere é formado por oito bytes, cada qual correspondendo ao padrão de uma das linhas que compõem a letra. Assim, cada oito bytes consecutivos na tabela de padrões correspondem a um caractere. O laço com o rótulo **ash** vai pulando de oito em oito bytes, até encontrar a posição do caractere que queremos imprimir. Para isso, utiliza A como contador. Em outras palavras, **add hl,de** soma 8 a HL e **dec a** subtrai um de A. A instrução **jr nz, ash** repete o processo até que A se reduza a zero. O endereço do byte inicial do padrão desejado estará, então, em HL. Este valor é transferido para a pilha e, em seguida, recuperado em BC. O valor original de HL — posição na tela — também é recuperado da pilha e a sub-rotina termina.

A sub-rotina **lprint** cuida da impressão do caractere. Ela começa guardando na pilha os pares AF, HL e BC. A continha o atributo do caractere, HL, a posição de impressão na tela, e BC, o endereço inicial do padrão do caractere. As linhas seguintes realizam uma série de testes com a posição de impressão. Ao final deles, DE contém a posição de impressão na tela.

O programa continua no rótulo **next**. Ali, **push de** coloca a posição de impressão na pilha e **ld de,16384** coloca em DE o endereço inicial da tela. O endereço da RAM em que vamos colocar o padrão do caractere é a soma da posição de impressão com o endereço inicial da tela — **add hl,de**. O resultado fica em HL. Recupera-se a posição de impressão em DE e o acumulador recebe o valor 8. BC recupera o endereço inicial do padrão e AF é colocado na pilha, onde A servirá de contador.

O rótulo **rept** representa o laço que desenha a letra. A instrução **ld a,(bc)** coloca em A o byte que contém uma linha do caractere e **ld (hl),a** transfere este mesmo valor para a tela. BC aponta para a tabela de padrões, enquanto HL aponta para a tela. A instrução **inc h** soma uma unidade ao registro H, o que equivale a somar 256 a HL. O valor resultante aponta para o byte imediatamente inferior ao anterior, de maneira que o caractere é desenhado linha por linha.

BC também tem seu conteúdo aumentado em uma unidade por **inc bc**. O contador do laço é recuperado da pilha — **pop af** — e A é diminuído em uma unidade. Se A não for igual a zero, a instrução **jr z,exit** é ignorada, AF volta para a pilha e o processador retorna para desenhar mais uma linha do caractere.

Como A continha inicialmente o valor 8, serão desenhadas exatamente oito linhas, uma abaixo da outra, completando uma letra. Quando A contém zero, **jr z,exit** desvia o programa para o rótulo **exit**. O atributo do caractere é, então, colocado na tabela de atributos. Esse parâmetro determina a cor do caractere. Recupera-se a posição de impressão em HL e o atributo vai para A — **pop hl** e **pop af**.

O conteúdo de DE volta para a pilha e este par recebe o endereço inicial da tabela de atributos — 22528. A instrução **add hl,de** calcula o endereço da RAM correspondente e o atributo é colocado nele por **ld (hl),a**. A posição de impressão na tela, preservada ora em DE, ora na pilha, volta para HL e a sub-rotina termina.

A sub-rotina **cl** limpa a tela, sendo chamada diversas vezes no decorrer do programa. Primeiro, o registro IX recebe o endereço inicial da tela — 16384. HL, por sua vez, recebe o número de bytes da mesma — 6912 —, servindo como contador. A instrução **ld a,0** coloca 0 em A. A instrução **ld (ix+0),a** coloca

0 na posição da tela apontada por IX, o que, efetivamente, limpa aquele byte. O registro IX, aumentado em uma unidade, passa a apontar para o próximo byte da tela. HL é diminuído em uma unidade e, enquanto não chegar ao valor zero, a instrução **jr nz,clp** retornará para apagar o próximo byte da tela. Na realidade, existem quatro linhas entre **dec hl** e **jr nz,clp**; contudo, elas só têm a função de prolongar o tempo, como o rótulo **ldq** da rotina anterior.



A primeira coisa a programar é uma página anunciando o videogame. As linhas seguintes criam essa página, obtendo os dados necessários em uma tabela, que montaremos mais tarde. Por enquanto, carregue o Assembler de INPUT com as seguintes modificações, para que funcione de maneira satisfatória: na linha 5120, troque o número 6 por 22; na linha 6110, adicione **:GOSUB 6150**; na linha 6150, troque **GOSUB 6260** por **GOSUB 6160**. Modifique a linha 5000 com **CLEAR 1500,&CFFF**, para proteger o topo da memória. Digite, em seguida, este programa:

```
10    org -12288
20    call 108
30    ld de,296
40    ld hl,-14000
50    ld bc,9
60    call 92
70    ld de,493
80    ld hl, 13991
90    ld bc,15
100    call 92
110    ld de,572
120    ld hl, 13976
130    ld bc,18
140    call 92
150    ld b,10
160    ldp ld hl, 65000
170    ld de,0
180    ldq dec hl
190    push hl
200    sbc hl,de
210    pop hl
220    jr nz,ldq
230    djnz ldp
240    ret
250    end
```

Grave o programa-fonte e depois monte-o. O programa em código pode ser gravado depois disso.

OS CÓDIGOS

O programa liga a tela de texto de quarenta colunas por intermédio de uma rotina da ROM cujo endereço é 108, em decimal. Em seguida, reserva uma porção da memória para os códigos ASCII do texto a ser impresso. Ela começa no endereço -14000.

Uma outra rotina da ROM, que fica no endereço 92, encarrega-se da transferência de uma área da RAM para a VRAM. Para que o texto certo seja colocado na posição desejada, certos parâmetros devem ser fornecidos.

O par DE precisa conter a posição inicial da área da VRAM em que vamos colocar o texto. O par HL, por sua vez, contém o endereço inicial da área da RAM que será transferida. BC, finalmente, funciona como contador de bytes, determinando que comprimento terá o texto que vai ser impresso.

Essa rotina da ROM é chamada três vezes, uma para cada linha que aparece na tela inicial. Antes de cada comando **call**, os valores adequados são colocados nos pares DE, HL e BC.

PROGRAMAÇÃO DE PAUSAS

Para que o jogador possa ler o título, devemos incluir uma pausa no programa. Colocamos 10 no registro B, de modo que o laço responsável pela pausa seja executado dez vezes.

O número 65000 é colocado em HL, cujo conteúdo diminui em uma unidade a cada volta do laço interno — rotulado com **ldq**. HL é guardado e recuperado da pilha só para ganharmos tempo.

Pode parecer estranho subtrair zero — conteúdo de DE — de HL a cada volta do laço. Mas trata-se apenas de um recurso utilizado para afetar o indicador de zeros, que não reage à instrução **pop**. O comando **jr nz** depende desse indicador para que o programa possa sair do laço. Quando HL for reduzido a zero e **sub hl, de** subtrair zero dele, o resultado será igual a zero; ativado o indicador de zeros, o processador sairá do laço.

Quando o programa estiver completo, ele prosseguirá imprimindo a página destinada à exibição das instruções. Mas, por enquanto, a instrução **ret** provoca o retorno ao BASIC.

O PROGRAMA BASIC

As linhas seguintes geram a tabela ASCII necessária ao funcionamento do programa em código.

```
5 CLS CLEAR 100, 14000
10 A$ "AVALANCHECriação  A  Doe
Programa  R  Neder"
20 FOR I=0 TO 4.
30 POKE  .4000+I.ASC(MID$(A$,I+
1,1))
40 NEXT
50 DEFUSR. -12288
60 X USR1(0)
70 CLS
80 END
```

A linha 5 limpa a tela e protege o topo da memória que conterá a tabela.

A linha 10 coloca o texto utilizado na página inicial dentro da variável **A$**. É, sem dúvida, bem melhor digitar uma tabela na forma de um texto do que digitar os números de todos os códigos ASCII correspondentes.





Em seguida, um laço **FOR...NEXT** coloca os códigos ASCII na memória, criando a tabela ASCII.

O próprio programa testa, então, a rotina em código — se ela estiver gravada na memória.

O jogo não vai precisar desse programa BASIC depois que a tabela tiver sido gravada. Guarde-o, porém, para ajudar na montagem.



A primeira coisa que se programa em qualquer jogo é uma tela exibindo o título. Embora a rotina de impressão esteja em código, será mais fácil colocar os códigos ASCII das letras na memória usando um programa BASIC:

```
1 CLEAR 200,16999
10 AD=17000
30 READ A$
40 FOR A=1 TO LEN(A$):B=ASC(MID
$(A$,A,1))
50 IF B<&H61 THEN POKE AD,B ELS
E POKE AD,B-96
60 AD=AD+1
70 NEXT A
80 DATA "avalanchecriacao: a.do
e programa: s. kellaway e g. he
adley"
```

Quando ele é executado, a tabela necessária ao jogo é criada na memória do microcomputador. Para gravar a tabela, utilize **CSAVEM "DADOS", 17000, 17059, 19000**.

Se quiser usar o Assembler de INPUT, precisará ampliar a memória disponível; se não o fizer, o espaço será insuficiente para o Assembler e o programa em código. Digite, então:

```
POKE 25,6
POKE 26,1
NEW
```

Depois, digite **CLEAR 200,18999** para proteger o topo da memória e, em seguida, o programa:

```
10   ORG 19000
20   JSR CLS
30   LDX #1066
40   LDY #17000
50   LDB #9
60   JSR LPRINT
70   LDX #1384
80   LDB #15
90   JSR LPRINT
100   LDX #1445
110   LDB #21
120   JSR LPRINT
130   LDX #1481
140   LDB #12
150   JSR LPRINT
160   LDA #5
170   PAUSE LDX #65535
180   PAUSE1 LEAX -1,X
190   BNE PAUSE1
200   DECA
210   BNE PAUSE
220   JSR CLS
230   RTS
240   LPRINT EQU 19174
250   CLS EQU 19148
```

Os mnemônicos podem ser gravados por meio da opção correspondente do Assembler. Monte o programa e grave-o em código — se não souber fazer isto, utilize o monitor Assembler.

Além desse programa e da tabela, as outras rotinas aqui apresentadas serão necessárias para que você veja o resultado por meio de **EXEC 19000**.

O PROGRAMA BASIC

O programa em BASIC começa reservando uma boa quantidade de memória para armazenar o programa em código. Em seguida, as letras utilizadas na primeira tela são colocadas na memória, representadas por seus códigos ASCII. Em geral, é preciso subtrair 96 de cada código para que eles correspondam aos códigos de letras invertidas. Algumas letras, porém — códigos menores que 61, em hexadecimal —, não precisam dessa operação.

LIMPEZA DA TELA

A rotina em código começa em 19000, depois da tabela ASCII, e salta para a sub-rotina **CLS**, que começa em 19148. Na primeira rotina, o endereço desse rótulo é definido por uma instrução **EQU**, para que os desvios sejam calculados. Eis a sub-rotina:

```
10   ORG 19148
20   CLS LDX #1024
30   LDA #128
40   CLS1 STA ,X+
50   CMPX #1536
60   BLO CLS1
70   RTS
```

O endereço inicial da tela — 1024 é colocado no registro X e o código do espaço em branco — 128 —, em A.

STA,X+ coloca esse código na posição apontada por X; o conteúdo de X aumenta em uma unidade. A rotina é repetida várias vezes até que X seja maior que 1536, endereço final da tela.

Como o conteúdo de X aumenta antes da colocação de 128 em um endereço da tela, um comando **BLO** — desvio se for "menor que" — e usado para sair do laço **CLSI**. Essa rotina deve ser gravada separadamente.

IMPRESSÃO NA TELA

Quando o microprocessador retorna da sub-rotina **CLS**, a rotina principal prepara os registros para que se faça a impressão na tela.

O registro X contém a posição que desejamos para o primeiro caractere da linha a ser impressa. O "A" de *Avalanche* será impresso, então, em 1066.

O registro Y contém a posição de memória da tabela ASCII da primeira letra a ser impressa. Assim, 17000 é colocado em Y. O registro B contém o número de caracteres a ser impresso. Como vamos começar escrevendo "AVALANCHE", 9 é colocado em B. Em seguida, ocorre um salto para a sub-rotina **LPRINT**. Mais uma vez uma instrução **EQU** define o endereço 19174 para efeito de cálculos de desvios.

A ROTINA LPRINT

Esta rotina começa em 19174. Sua função consiste em obter os dados da tabela ASCII, colocando os códigos na tela, um de cada vez.

```
10  ORG 19174
20  LPRINT LDA ,Y+
30  STA ,X+
40  DECB
50  BNE LPRINT
60  RTS
```

Os códigos da tabela apontados por Y são colocados em A e o apontador volta-se para a próxima posição. O comando **STA,X+** coloca-os, então, na posição de tela apontada por X, e X aumenta em uma unidade, apontando para a posição seguinte.

A instrução **DECB** subtrai uma unidade do conteúdo de B e a rotina é repetida até que B chegue a zero. Quando isso ocorre, o microprocessador retorna à rotina principal. Guarde esta rotina separadamente.

A rotina prossegue até completar a página de rosto de nosso videogame, imprimindo linha por linha.

Para escrever uma nova linha, a posição de impressão é colocada em X e o comprimento do texto, em B.

Não há necessidade de colocarmos um novo valor em Y, já que esse apontador simplesmente percorre a tabela ASCII B caracteres de cada vez.

A ROTINA DE PAUSA

Quando a última linha da página inicial tiver sido impressa, o microprocessador fará uma pausa no programa, para que possamos lê-la, antes de ser substituída pela tela de instruções.

O número 5 é colocado no acumulador e o registro X recebe o valor 65535. **LEAX -1,X** subtrai uma unidade de X e **BNE PAUSEI** retorna, de modo que vai diminuindo até ser reduzido a zero; A também é diminuído. O processo se repete até que A também seja igual a zero e o comando **BNE** permita que o processador deixe o laço.

DOIS LAÇOS

O laço externo **PAUSE** — baseado no acumulador — repete-se, assim, cinco vezes; a cada repetição, o laço interno **PAUSEI** é executado 65535 vezes. O uso de dois laços tem a vantagem de possibilitar o acerto do tempo. O laço externo nos dá um ajuste grosseiro e o laço interno, um ajuste mais fino.

Por fim, a rotina **CLS** limpa a tela novamente. Em seguida, o computador passa à página de instruções, que apresentaremos no próximo artigo desta série. Por hora, uma instrução **RTS** retorna ao BASIC



O JOGO DO OTELO (1)

Embora suas regras sejam bastante simples, o jogo do Otelo requer muita estratégia. Programá-lo é fácil: monte o tabuleiro, crie as peças e desafie sua máquina!

Otelo é um jogo de estratégia que se desenvolve num tabuleiro de oito por oito posições — como os tabuleiros de xadrez ou dama. As regras são bem fáceis e o jogo também.

O objetivo é tomar o maior número possível de peças do oponente. A jogada consiste basicamente na colocação das peças no tabuleiro, uma a uma, por um jogador de cada vez, até preenchê-lo totalmente. Cada jogador começa com duas peças e tenta tomar as do oponente, "cercando-as". Para consegui-lo, deve colocar as peças de maneira que as do oponente fiquem entre as suas — ou seja, que se forme uma trilha de peças adversárias com peças suas nas duas extremidades. Ao tomar uma trilha adversária, esteja ela na horizontal, vertical ou diagonal, todas as peças do oponente serão substituídas por peças do jogador em questão.

O placar de cada jogador corresponde ao número de peças que ele tem no tabuleiro naquele momento. Será vencedor aquele que tiver o maior número de peças no tabuleiro quando ele estiver totalmente cheio.

Nesta versão para computador, estaremos jogando contra a máquina que, além de mostrar o tabuleiro, também atualiza o placar.

DICAS PARA VENCER

Para quem nunca jogou Otelo, as dicas que se seguem serão muito úteis.

As posições dos cantos são extremamente importantes, já que, uma vez conquistadas, jamais serão roubadas. Em conseqüência, a ocupação de tais posições é fundamental para garantir a vitória — mesmo que implique o sacrifício de algumas de nossas peças ou impeça um outro movimento que pudesse nos render mais peças do oponente.

Se considerarmos que uma peça pode completar mais de uma linha — na vertical, na horizontal e na diagonal —, concluiremos que o movimento aparentemente mais óbvio nem sempre é o melhor, pois, nos estágios mais avançados do jogo, será possível tomar duas ou três linhas de peças do oponente com a simples adição de uma peça.

Pense sempre adiante. Podemos induzir o adversário a criar situações que nos ajudem a tomar posições vitais, simplesmente simulando ou fingindo que fizemos uma má jogada.

O PROGRAMA

Uma das grandes vantagens da versão de Otelo para computador sobre o verdadeiro tabuleiro é que nela o computador fica com todo o "trabalho pesado". Além de desempenhar o papel de adversário, ele substitui as peças conquistadas e cuida do placar, deixando-nos livres para pensar só nas jogadas.

O computador demora um pouco para fazer suas jogadas, pois leva em conta todos os movimentos possíveis. No decorrer do jogo, à medida que o número de posições vazias diminui, passa a tomar decisões com maior rapidez.

JOGANDO

Ao rodar o programa, o computador perguntará se você deseja fazer a primeira jogada ou não. Para executar uma jogada, é necessário fornecer-lhe duas coordenadas. Estas especificam a posição da peça no tabuleiro e estão compreendidas entre os valores 1 e 8. Fornecemos primeiramente a coordenada da linha — ou seja, do eixo Y, com origem no topo da tela — e, depois, a coordenada da coluna — eixo X, com origem no lado esquerdo do tabuleiro. Se fornecermos 0 (zero) para as coordenadas, o jogo será encerrado; se fornecermos 9, transferiremos o direito de jogar para o computador.

PRIMEIRAS LINHAS

Digite agora a primeira parte do jogo do Otelo. Se você rodar o programa como está, verá o tabuleiro na tela, mas ainda não poderá jogar.

Salve as linhas digitadas em disco ou em fita cassete para mais tarde incorporá-las à segunda parte do programa, que será apresentada num próximo artigo, completando o jogo.

S

```
10 BORDER 0: PAPER 7: INK 1
CLS
15 PRINT AT 1,11; INVERSE 1;"
OTHELLO"
20 PRINT AT 10,2;"VOCE QUER C
OMECAR ?  (S OU N)": INPUT X$
: IF X$="" THEN GOTO 20
30 LET X$=X$(1): IF X$<>"S"
AND X$<>"N" AND X$<>"s" AND X
$<>"n" THEN GOTO 20
40 LET CP=1: IF X$="N" OR X$=
"n" THEN LET CP=2
100 DIM B(8,8): DIM C(8): DIM
D(8,2): DIM X(60): DIM Y(60):
DIM N(60)
110 LET B(4,4)=1: LET B(4,5)=2
: LET B(5,4)=2: LET B(5,5)=1
120 FOR F=1 TO 8: READ A: LET
D(F,1)=A: READ A: LET D(F,2)=A
: NEXT F
130 DATA -1,-1,0,-1,1,-1,-1,0,
1,0,-1,1,0,1,1,1
140 FOR F=0 TO 7: READ A,B,C:
POKE USR "A"+F,A: POKE USR "B"
+F,B: POKE USR "C"+F,C: NEXT F
150 DATA 204,0,0,51,60,60,204,
126,66,51,126,66,204,126,66,51
,126,66,204,60,60,51,0,0
```



```
10  HOME : VTAB (3): HTAB (15)
15  INVERSE : PRINT " O T E L O
 ": NORMAL
20  VTAB (10): PRINT "QUER JOGA
R PRIMEIRO (S/N) ";: INPUT X$:
IF X$ = "" THEN 20
30 X$ =  LEFT$ (X$,1):CP = 1: I
F X$ = "N" THEN CP = 2
40  IF X$ <  > "S" AND X$ <  >
"N" THEN  GOTO 20
100  DIM B(8,8),C(8),D(8,2),X(6
0),Y(60),N(60)
110 B(4,4) = 1:B(4,5) = 2:B(5,4
) = 2:B(5,5) = 1
120  FOR F = 1 TO 8: READ A:D(F
,1) = A: READ A:D(F,2) = A: NEX
T
130  DATA  -1,-1,0,-1,1,-1,-1,0
,1,0,-1,1,0,1,1,1
```

MSX

```
5 COLOR 1,15,15:SCREEN2
10 FOR X=1 TO 16:READ A:NEXTX:G
OSUB 9200:RESTORE
```

- AS REGRAS DO JOGO
- DICAS PARA VENCER
- A PRIMEIRA TELA
- SELEÇÃO DOS GRÁFICOS PARA O TABULEIRO E AS PEÇAS
- INICIALIZAÇÃO DO JOGO
- DIGITANDO A ROTINA PRINCIPAL
- A VEZ DO JOGADOR

```
15 DRAW"BM60,40;S25;C1"·A$="
LO":GOSUB 9300
20 DRAW"BM26,120;S10;C2" A$="QU
ER JOGAR PRIMEIRO"·GOSUB 9300·D
RAW"BM90,140"·A$="S OU N" GOSUB
 9300
21 X$=INKEY$·IFX$=""THEN 21
25 COLOR15,4,4
30 IFX$<>"S"ANDX$<>"s"ANDX$<>"N
"ANDX$<>"n"THEN 21
40 CP=1:IFX$="N"ORX$="n"THEN CP
=2
100 DIM B(8,8),C(8),D(8,2),X(60
),Y(60),N(60)
110 B(4,4)=1 B(4,5)=2·B(5,4)=2·
B(5,5)=1
120 FOR F=1 TO 8·READ A D(F,1)=
A READ A·D(F,2)=A NEXTF
130 DATA -1,-1,0,-1,1,-1,-1,0,1
,0,-1,1,0,1,1,1
150 GOSUB 1100
```



```
10 PMODE 1,1·COLOR 4,1:PCLS·SCR
EEN 1,0·FOR X=1 TO 16·READ A:NE
XTX·GOSUB 9200:RESTORE
15 DRAW"BM60,40;S16":A$="OTHELL
O":GOSUB 9300
20 DRAW"BM26,120,S8,C2":A$="  Q
UER JOGAR PRIMEIRO":GOSUB 9300:
DRAW"BM100,140":A$="S OU N":GOS
UB 9300
21 X$=INKEY$:IF X$="" THEN 21
30 IF X$<>"S" AND X$<>"N" THEN
21
40 CP=1:IF X$="N" THEN CP=2
100 DIM B(8,8),C(8),D(8,2),X(60
),Y(60),N(60)
110 B(4,4)=1:B(4,5)=2:B(5,4)=2·
B(5,5)=1
120 FOR F=1 TO 8:READ A:D(F,1)=
A·READ A:D(F,2)=A·NEXT F
130 DATA -1,-1,0,-1,1,-1, 1,0,1
,0,-1,1,0,1,1,1
150 GOSUB 1100
```

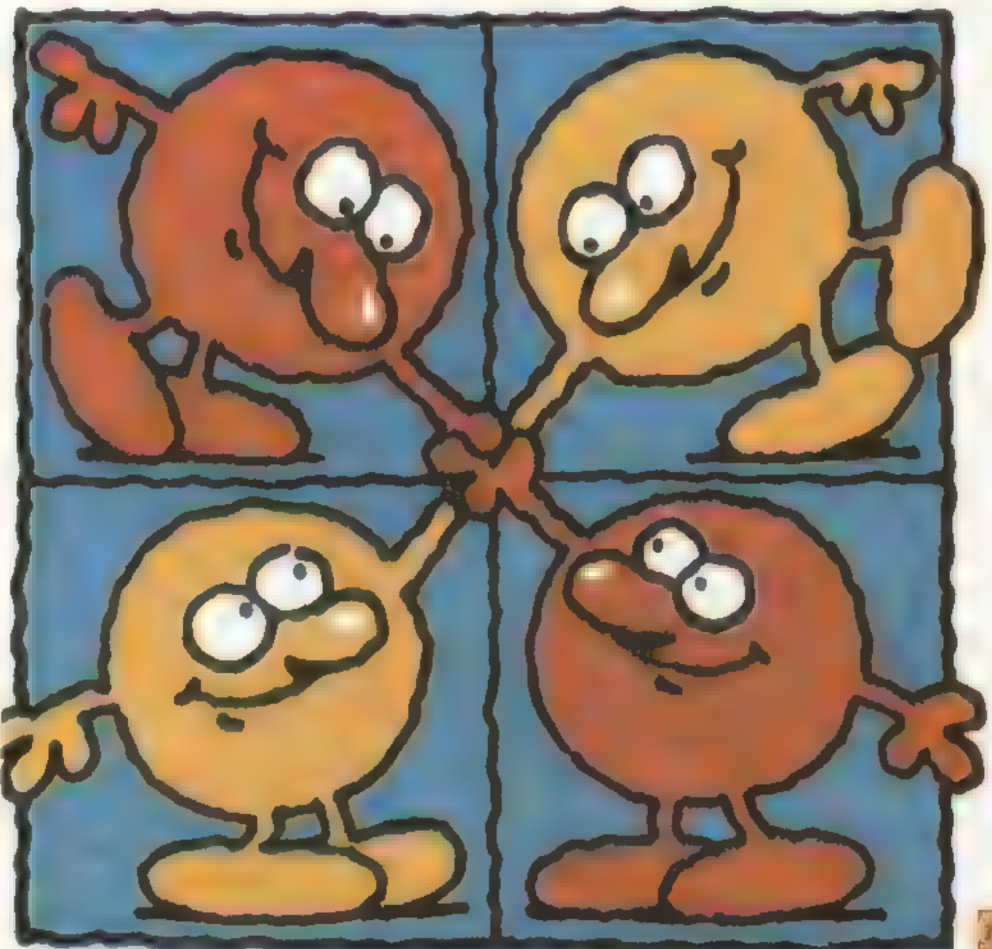
As linhas 10 a 40 geram a primeira tela que o jogador vê, ou seja, o título do jogo e a pergunta que o inicia. A linha 10 (linha 5, no MSX) se encarrega do ajustamento das cores e da limpeza da tela. O nome do jogo é gerado na linha 15. Como nos programas para os micros TRS-Color e MSX usou-se a tela de alta resolução, existe uma rotina que desenha letras no final do programa (a qual já utilizamos anteriormente em INPUT).

A linha 20 apresenta a primeira pergunta, destinada a definir quem jogará primeiro. A resposta, armazenada em **X$**, é depois reduzida à sua primeira letra, na linha 30. **CP** corresponde ao jogador em questão e recebe o valor 1 quando quem joga é você e 2 quando é o computador. A linha 40 faz a verificação por S e N.

As linhas 100 a 130 inicializam as variáveis e matrizes usadas no jogo. A linha 100 cuida das matrizes. **B(x,y)** representa o tabuleiro; os valores armazenados em cada elemento representam o status do quadrado correspondente no tabuleiro. Se o valor for 0, o quadrado está vazio; se for 1, uma peça pertencente ao jogador ocupa aquela posição; e se for 2, a posição está ocupada por uma peça do computador. **C(x)** é usado para verificar as jogadas do jogador. **D(x,y)** contém os deslocamentos X e Y nas oito direções possíveis da jogada. **X(x)**, **Y(x)** e **N(x)** são usadas nos cálculos para a jogada do computador.

A linha 110 especifica as posições iniciais no tabuleiro. Cada jogador tem duas peças posicionadas em seu centro, no começo da disputa. As direções possíveis, partindo dessa posição, são definidas pela linha **DATA** 130. Cada valor representa um deslocamento X ou Y, com os números negativos indicando esquerda ou topo do tabuleiro, respectivamente.

Na versão para os microcomputadores Spectrum, as linhas 140 e 150 geram UDG para o quadrado vazio e para as peças. A linha 140 procede à leitura dos dados apresentados pela linha 150 e cria os UDG **A**, **B** e **C**.



O LAÇO PRINCIPAL



```
500 GOSUB 1000
505 IF CS+PS=64 THEN  GOTO
 4000
510 LET EG=0  IF CP=1 THEN
GOSUB 2000  GOSUB 1000· IF EG=
1 THEN  GOTO 4000
515 IF CS+PS=64 THEN  GOTO
 4000
520 IF CP=2 THEN  GOSUB 3000
530 GOTO 500
```



```
500  GOSUB 1000
505  IF CS + PS = 64 THEN 4000
510 EG = 0: IF CP = 1 THEN  GOS
UB 2000· GOSUB 1000  IF EG = 1
THEN 4000
515  IF CS + PS = 64 THEN 4000
520  IF CP = 2 THEN  GOSUB 3000
530  GOTO 500
```



```
500 GOSUB 1000
505 IF CS+PS=64 THEN 4000
510 EG=0·IF CP=1 THEN GOSUB 200
0 GOSUB 1000:LINE(112,150)-(144
,180),1,BF·IF EG=1 THEN 4000
515 IF CS+PS=64 THEN 4000
520 IF CP=2 THEN GOSUB 3000
530 GOTO 500
```

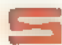


```
500 GOSUB 1000
505 IF CS+PS=64 THEN 4000
510 EG=0:IF CP=1 THEN GOSUB 200
0:GOSUB 1000:COLOR 1:LINE(118,1
50)-(150,180),PSET,BF.IF EG=1 T
HEN 4000
515 IF CS+PS=64 THEN 4000
520 IF CP=2 THEN GOSUB 3000
530 GOTO 500
```

O laço principal do programa está entre as linhas 500 e 530. A linha 500 chama a sub-rotina que desenha o tabuleiro. As linhas 505 e 515 verificam se o tabuleiro está cheio, ou seja, se a soma dos pontos do jogador, **PS**, com os do computador, **CS**, resulta em 64 (número de posições do tabuleiro).

A linha 510 ajusta o indicador de final de jogo **(EG)** para 0 e chama a rotina responsável pela vez do jogador, seguida pela sub-rotina do tabuleiro. Se depois da execução dessas sub-rotinas o indicador **EG** for setado (valor 1), o programa saltará para a sub-rotina de final de jogo, que começa na linha 4000. A jogada do computador é feita pela linha 520, caso **CP** seja igual a 2 (sub-rotina 3000).

DESENHANDO O TABULEIRO



```
1000 CLS · PRINT TAB 11;"123456
78"· LET PS=0: LET CS=0
1010 FOR F=1 TO 8  PRINT TAB 9;
F," ";· FOR G=1 TO 8
1020 IF B(F,G)=0 THEN  PRINT CH
R$ 144;
1030 IF B(F,G)=1 THEN  PRINT  I
NK 2;CHR$ 145;  LET PS=PS+1
1040 IF B(F,G)=2 THEN  PRINT  I
NK 2,CHR$ 146,· LET CS=CS+1
1050 NEXT G  PRINT · NEXT F
1052 LET P$="PONTOS"· IF PS=1 T
HEN  LET P$="PONTO"
1054 LET Q$="PONTOS"· IF CS=1 T
HEN  LET Q$="PONTO"
1060 PRINT '' INK 2;" VOCE =";T
AB 20;"COMPUTADOR="  PRINT " ",
PS;" ",PS;TAB 20;CS;" ";Q$
1070 RETURN
```



```
1000  HOME  PRINT  TAB( 18):"1
2345678"·PS = 0·CS = 0· PRINT
1010  FOR F = 1 TO 8: PRINT  TA
B( 16);F;" ";· FOR G = 1 TO 8
1015  IF B(F,G) <  > 1 AND B(F,
G) <  > 2 THEN B(F,G) = 0
1020  IF B(F,G) = 0 THEN  PRINT
 "+";
1030  IF B(F,G) = 1 THEN  PRINT
  CHR$ (48);·PS = PS + 1
1040  IF B(F,G) = 2 THEN  INVER
SE   PRINT " ",· NORMAL ·CS = C
S + 1
1050  NEXT G· PRINT · NEXT F
1052 P1$ = " PONTOS": IF PS = 1
 THEN P1$ = " PONTO"
1054 P2$ = " PONTOS": IF CS = 1
 THEN P2$ = " PONTO"
1060  PRINT   PRINT "JOGADOR =
0"; TAB( 25),"COMPUTADOR = ";·
INVERSE : PRINT " ": NORMAL : P
RINT PS;P1$; TAB( 25);CS;P2$
1070  RETURN
```



```
1000 PS=0:CS=0
1010 FOR F=1TO8:FOR G=1TO8
1030 IF B(F,G)=1 THEN LINE((G-1
)*16+64,(F-1)*16+12) (G*16+64,F
*16+12),15,BF·PS=PS+1
1040 IF B(F,G)=2 THEN LINE((G 1
)*16+64,(F 1)*16+12)-(G*16+64,F
*16+12),1,BF:CS=CS+1
1050 NEXTG,F
1060 DRAW"BM10,90:C1" A$="VOCE"
·GOSUB9300·DRAW"BM200,90":A$="C
OMPUT"·GOSUB9300
1063 LINE(20,110)-(28,118),15,B
F:LINE(220,110)-(228,118),1,BF
1065 LINE(5,60) (42,80),1,BF LI
NE(205,60)-(242,80),1,BF
1066 DRAW"BM12,62;S16;C15"·A$ M
ID$(STR$(PS),2)·GOSUB9300 DRAW"
BM212,62"·A$=MID$(STR$(CS),2) G
OSUB9300·DRAW"S8"
1070 RETURN
1100 CLS·FORX=0TO7 DRAW"BM"+STR
$(X*16+70)+",0;S8"+NU$(X+1) NEX
TX
1110 LINE(64,12)-(192,140),2,BF
1120 FORX=0TO8·LINE(X*16+64,12)
-(X*16+64,140),1·NEXT
1130 FORY=0TO8·LINE(64,Y*16+12)
 (192,Y*16+12),1 NEXT
1140 FORY=0TO7:DRAW"C15S8BM54,"
+STR$(Y*16+14)+NU$(Y+1)·NEXT
1150 RETURN
```



```
1000 PS=0:CS=0
1010 FOR F=1 TO 8:FOR G=1 TO 8
1030 IF B(F,G)=1 THEN PAINT(G*1
6+54,F*16),1,3:PS=PS+1
1040 IF B(F,G)=2 THEN PAINT(G*1
6+54,F*16),4,3:CS=CS+1
1050 NEXT G,F
1060 DRAW"BM0,150;C2":A$=" VOCE
 ":GOSUB 9300:DRAW"BM186,150":
A$="COMPUT":GOSUB 9300
1063 LINE(62,150) (70,158),PSET
,B:LINE(246,150)-(256,158),PSET
,B
1064 PAINT(248,152),4,2
1065 COLOR 1·LINE(0,60)-(32,80)
,PSET,BF:LINE(216,60)-(248,80),
PSET,BF
1066 DRAW"BM0,60;S16;C4":A$=MID
$(STR$(PS),2)·GOSUB 9300:DRAW"B
M216,60" A$=MID$(STR$(CS),2)·GO
SUB 9300:DRAW"S8"
1070 RETURN
1100 PCLS1:COLOR2:FORX=0 TO 7.D
RAW"BM"+STR$(X*16+70)+",0;S8"+N
U$(X+1):NEXT X
1110 LINE(64,12)-(192,140),PSET
,BF:COLOR 3
1120 FOR X=0 TO 8:LINE(X*16+64,
12)-(X*16+64,140),PSET:NEXT
1130 FOR Y=0 TO 8·LINE(64,Y*16+
12)-(192,Y*16+12),PSET:NEXT
1140 FOR Y=0 TO 7·DRAW"C2S8BM54
,"+STR$(Y*16+14)+NU$(Y+1)·NEXT
1150 RETURN
```

A sub-rotina que tem início na linha 1000 desenha o tabuleiro. Primeiro, limpa a tela, imprime (ou desenha, no caso dos microcomputadores das linhas MSX e TRS-Color) os números das colunas e zera os dois placares.

O laço **FOR...NEXT** que fica entre as linhas 1010 e 1050 do programa desenha o tabuleiro na tela, linha por linha. À medida que as peças são mostradas, incrementa-se o placar.

As linhas 1052 e 1054 melhoram a "gramática" do placar, definindo se a palavra PONTO será usada no singular

ou no plural. A linha 1060, por sua vez, mostra o placar para o jogador.

No micro TRS-Color e no MSX, as linhas 1100 a 1150 constituem comandos extras de alta resolução para o desenho do tabuleiro.

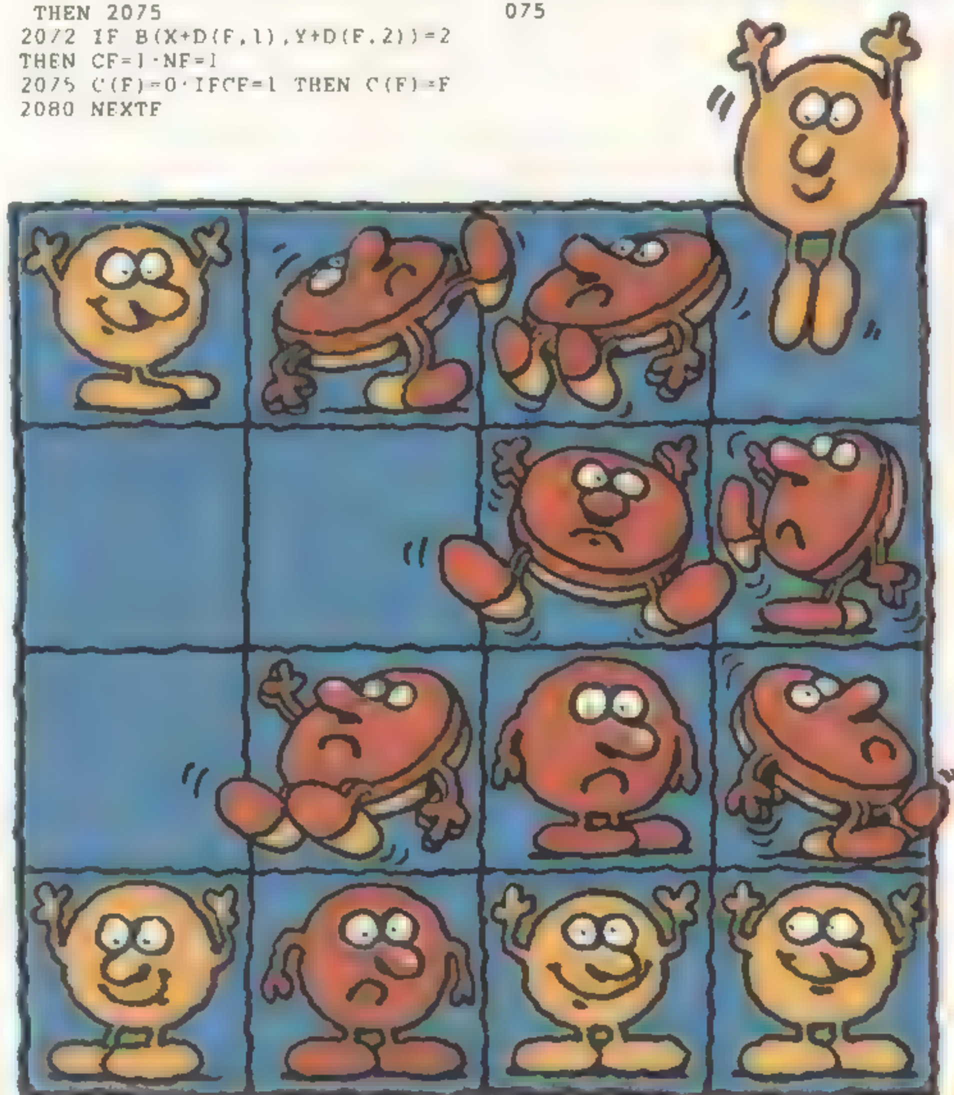
A VEZ DO JOGADOR

```
2000 PRINT AT 14,0;"FACA O MOVIMENTO (LINHA,COLUNA)": INPUT X,Y
2005 IF X=0 THEN  LET EG=1: RETURN
2006 IF X=9 THEN  LET CP=2: RETURN
2010 IF X<1 OR X>8 OR Y<1 OR Y>8 THEN  GOTO 2000
2020 IF B(X,Y)=0 THEN  GOTO 2070
2040 PRINT AT 17,0;" VOCE NAO PODE IR PARA UMA CASA  OCUPADA!"
  FOR F=1 TO 500: NEXT F
2050 PRINT AT 17,0;"                                "
: GOTO 2000
2070 LET NF=0: FOR F=1 TO 8: LET CF=0: IF X+D(F,1)=0 OR X+D(F,1)=9 THEN  GOTO 2075
2071 IF Y+D(F,2)=0 OR Y+D(F,2)=9 THEN  GOTO 2075
2072 IF B(X+D(F,1),Y+D(F,2))=2 THEN  LET CF=1: LET NF=1
2075 LET C(F)=0: IF CF=1 THEN LET C(F)=F
2080 NEXT F
2090 STOP
```

```
2000  VTAB (15): INPUT "QUAL E SUA JOGADA (LINHA,COLUNA) ? ";X,Y
2005  IF X = 0 THEN EG = 1: RETURN
2006  IF X = 9 THEN CP = 2: RETURN
2010  IF (X < 1 OR X > 8) OR (Y < 1 OR Y > 8) THEN 2000
2020  IF B(X,Y) = 0 THEN 2070
2040  VTAB (17): PRINT "ESTE LUGAR JA' ESTA' OCUPADO": FOR F = 0 TO 1500: NEXT
2050  VTAB (17): PRINT "                              ": GOTO 2000
2070 NF = 0: FOR F = 1 TO 8:CF = 0: IF X + D(F,1) = 9 OR X + D(F,1) = 0 THEN 2075
2071  IF Y + D(F,2) = 9 OR Y + D(F,2) = 0 THEN 2075
2072  IF B(X + D(F,1),Y + D(F,2)) = 2 THEN CF = 1:NF = 1
2075 C(F) = 0: IF CF = 1 THEN C(F) = F
2080  NEXT F
2090  STOP
```

```
2000 LINE(0,182)-(255,191),1,BF:LINE(112,150)-(144,180),1,BF:DRAW"C15;BM10,182":A$="SUA JOGADA LINHA E COLUNA":GOSUB9300
2001 I$=INKEY$:IFI$<"0" OR I$>"9"THEN2001
2002 X=VAL(I$):DRAW"BM114,154;S16,C15"+NU$(X)
2003 I$=INKEY$:IFI$<"0" OR I$>"9"THEN2003
2004 Y=VAL(I$):DRAWNU$(Y)+"S8"
2005 IFX=0 THEN EG=1:RETURN
2006 IFX=9 THEN CP=2:RETURN
2010 IFX<1 OR X>8 OR Y<1 OR Y>8 THEN 2000
2020 IF B(X,Y)=0 THEN 2070
2040 LINE(0,182)-(255,191),1,BF:DRAW"S8C15BM50,182":A$="QUADRADO OCUPADO":GOSUB9300:FORF=1TO900:NEXTF
2050 GOTO 2000
2070 NF=0:FORF=1TO8:CF=0:IFX+D(F,1)=0 OR X+D(F,1)=9 THEN 2075
2071 IFY+D(F,2)=0 OR Y+D(F,2)=9 THEN 2075
2072 IF B(X+D(F,1),Y+D(F,2))=2 THEN CF=1:NF=1
2075 C(F)=0:IFCF=1 THEN C(F)=F
2080 NEXTF
```

```
2000 COLOR 1:LINE(0,182)-(255,191),PSET,BF:LINE(118,150)-(150,180),PSET,BF:DRAW"C3;BM0,182":A$="PARA ONDE     LINHA E COLUNA":GOSUB 9300:SOUND 100,1
2001 I$=INKEY$:IF I$<"0" OR I$>"9" THEN 2001
2002 X=VAL(I$):DRAW"BM118,150;S16;C4"+NU$(X)
2003 I$=INKEY$:IF I$<"0" OR I$>"9" THEN 2003
2004 Y=VAL(I$):DRAW NU$(Y)+"S8"
2005 IF X=0 THEN EG=1:RETURN
2006 IF X=9 THEN CP=2:RETURN
2010 IF X<1 OR X>8 OR Y<1 OR Y>8 THEN 2000
2020 IF B(X,Y)=0 THEN 2070
2040 COLOR 1:LINE (0,182)-(255,191),PSET,BF:DRAW"S8C4BM0,182":A$="VOCE NAO PODE IR PARA AI":GOSUB 9300:FOR F=1 TO 900:NEXT F
2050 GOTO 2000
2070 NF=0:FOR F=1 TO 8:CF=0:IF X+D(F,1)=0 OR X+D(F,1)=9 THEN 2075
```



```
2071 IF Y+D(F,2)=0 OR Y+D(F,2)=
9 THEN 2075
2072 IF B(X+D(F,1),Y+D(F,2))=2
THEN CF=1:NF=1
2075 C(F)=0:IF CF=1 THEN C(F)=F
2080 NEXT F
```

Sua jogada está entre as linhas 2000 e 2270, mas, por enquanto, o programa vai somente até a linha 2090 (2080, no MSX e no TRS-Color).

O programa obtém as coordenadas X e Y do jogador na linha 2000. A linha 2005 verifica se X é zero; se for, o indicador de fim de jogo é setado.

A linha 2010 verifica se houve um eventual erro de entrada das coordenadas do jogador; se houve, volta para a linha 2000. O programa checa, então, o quadrado escolhido, para se certificar de que está vazio, e salta para a linha 2070. Caso o quadrado já esteja ocupado por outra peça, a linha 2040 imprime uma mensagem de erro.

As linhas finais desta parte do programa (2070 a 2075) verificam se a peça em questão está ou não próxima de uma das peças do computador.

ROTINA PARA DESENHAR LETRAS

Para que possamos escrever na tela de alta resolução do MSX e do TRS-Color, precisaremos acrescentar esta rotina. Ela é bem semelhante à que foi descrita no artigo da página 232.



```
9100 DATA BR4,ND4R3D2NL3ND2BE2,
ND4R3DGNL2FDNL3BU4BR2,NR3D4R3BU
4BR2,ND4R2FD2GL2BE4BR,NR3D2NR2D
2R3BU4BR2
9110 DATA NR3D2NR2D2BE4BR,NR3D4
R3U2LBE2BR,D4BR3U2NL3U2BR2,ND4B
R2,BD4REU3L2R3BR2,D2ND2NF2E2BR2
9120 DATA D4R3BU4BR2,ND4FREND4B
R2,ND4F3DU4BR2,NR3D4R3U4BR2,ND4
R3D2NL3BE2,NR3D4R3NHU4BR2
9130 DATA ND4R3D2L2F2BU4BR2,BD4
R3U2L3U2R3BR2,RND4RBR2,D4R2U4BR
2,D3FEU3BR2,D4EFU4BR2
9140 DATA DF2DBL2UE2UBR2,DFND2E
UBR2,R3G3DR3BU4BR2
9150 DATA NR2D4R2U4BR2,BDEND4BR
2,R2D2L2D2R2BU4BR2,NR2BD2NR2BD2
R2U4BR2,D2R2D2U4BR2,NR2D2R2D2L2
BE4,D4R2U2L2BE2BR2,R2ND4BR2,NR2
D4R2U2NL2U2BR2,NR2D2R2D2U4BR2
9200 DIM LE$(26)
9210 FOR K=0 TO 26:READ LE$(K)
NEXT
9220 FOR K=0 TO 9:READ NU$(K):N
EXT
9230 RETURN
9300 FOR K=1 TO LEN(A$)
9310 B$=MID$(A$,K,1)
9320 IF B$>="0" AND B$<="9" THE
N DRAW NU$(VAL(B$)) GOTO 9350
9330 IF B$=" " THEN N=0 ELSE N=
ASC(B$)-64
9340 DRAW LE$(N)
9350 NEXT
9360 RETURN
```



```
9100 DATA BR4,ND4R3D2NL3ND2BE2
ND4R3DGNL2FDNL3BU4BR2,NR3D4R3BU
4BR2,ND4R2FD2GL2BE4BR,NR3D2NR2D
2R3BU4BR2
9110 DATA NR3D2NR2D2BE4BR,NR3D4
R3U2LBE2BR,D4BR3U2NL3U2BR2,ND4B
R2,BD4REU3L2R3BR2,D2ND2NF2E2BR2
9120 DATA D4R3BU4BR2,ND4FREND4B
R2 ND4F3DU4BR2,NR3D4R3U4BR2,ND4
R3D2NL3BE2,NR3D4R3NHU4BR2
9130 DATA ND4R3D2L2F2BU4BR2,BD4
R3U2L3U2R3BR2,RND4RBR2,D4R2U4BR
2,D3FEU3BR2,D4EFU4BR2
9140 DATA DF2DBL2UE2UBR2,DFND2E
UBR2,R3G3DR3BU4BR2
9150 DATA NR2D4R2U4BR2,BDEND4BR
2,R2D2L2D2R2BU4BR2,NR2BD2NR2BD2
R2U4BR2,D2R2D2U4BR2,NR2D2R2D2L2
BE4,D4R2U2L2BE2BR2,R2ND4BR2,NR2
D4R2U2NL2U2BR2,NR2D2R2D2U4BR2
9200 DIM LE$(27)
9210 FOR K=0 TO 76 READ LE$ K
NEXT
9220 FOR K=0 TO 9:READ NU$(K):N
EXT
9230 RETURN
9300 FOR K=1 TO LEN(A$)
9310 B$=MID$(A$,K,1)
9320 IF B$>="0" AND B$< "9" THE
N DRAW NU$(VAL(B$)) GOTO 9350
9330 IF B$=" " THEN N=0 ELSE N=
ASC(B$) 64
9340 DRAW LE$(N)
9350 NEXT
9360 RETURN
```



AS INSTRUÇÕES DO JOGO

- MAIS TABELAS DE DADOS EM ASCII
- ESCREVA NA TELA
- COMO CHAMAR SUB-ROTINAS DA ROM

Impressas na tela, as informações contidas no programa deste artigo permitirão que ajudemos o infortunado Willie em sua incansável busca do lanche roubado.

A primeira coisa a fazer é colocar o título na tela. Em seguida, são fornecidas as instruções. Como estamos programando e conhecemos cada detalhe envolvido, às vezes esquecemos que outras pessoas não têm a menor idéia dos objetivos do jogo e não sabem como jogar. Assim, é necessário colocar na tela toda informação relevante, de maneira que o usuário não seja obrigado a parar o jogo a cada momento para ler o folheto de instruções. As informações devem ser bem claras e objetivas. Ao mesmo tempo, é conveniente "temperar" o texto com algum enredo, já que um videogame deve apelar à imaginação do jogador e não simplesmente testar seus reflexos.



Mais uma vez, utilizaremos um programa em BASIC para colocar as instruções — letra por letra — na memória, criando uma tabela de códigos ASCII.

Para que a tabela sirva aos nossos propósitos, é importante que o numero de códigos de caracteres transferidos para a memória seja exato — afinal, eles vão ser lidos pelo programa em código. Como o leitor pode ter problemas para descobrir o número de espaços em cada linha, vamos conferir: no final da linha 100, há um espaço; no final da linha 120, três; na 140, três; na 160, catorze; na 180, um; na 240, dois; na 300, três, e na 320, 26 espaços em branco. Nas linhas 330 e 340 há quatro espaços entre as teclas e o hífen, e um espaço entre o hífen e a função da tecla. No final da linha 330 temos dezenove espaços e no final da 340, dezenove espaços também. Todos os demais claros na listagem correspondem a um caractere de espaço apenas. Outra maneira de conferir os dados consiste em contar exatamente 32 caracteres em cada linha **DATA**, exceto na 320, que tem 34 caracteres.

Antes de passar à execução, proteja o topo da memória com **CLEAR 57434**.

Quando executarmos o programa, será criada uma tabela com instruções. Ao

final do programa, o laço **FOR...NEXT** entre as linhas 500 e 520 projeta na tela uma imagem da porção da memória que contém a tabela.

```
 10 LET x=57480
 20 FOR n=1 TO 16
 30 READ a$: FOR o=1 TO LEN a$
 : POKE x,CODE (a$(o TO o))
 : LET x=x+1: NEXT o
 40 NEXT n
100 DATA " Apos um pequeno pas
seio,Willie "
120 DATA "volta e descobre que
 um bando        "
140 DATA "de cabritos espalhou
 todo seu       "
160 DATA "lanche na encosta
               "
180 DATA "Agora ele deve subir
 ao topo da "
200 DATA "montanha e recuperar
 suas coisas"
220 DATA "enfrentando uma aval
anche,cobras"
240 DATA "e podendo ate cair n
um buraco        "
260 DATA "Para piorar a situac
ao a mare        "
280 DATA "esta subindo e ele c
orre o risco"
290 DATA "de se afogar. Para a
judar Willie"
300 DATA "leia as instrucoes e
 aperte a       "
320 DATA "tecla 'S'
                 "
330 DATA "N      - Correr
         "
340 DATA "M        Saltar
         "
360 DATA "  ou Ambos"
500 FOR n=57435 TO 58000
510 PRINT CHR$ PEEK n;
520 NEXT n
```

Os dados que este programa coloca na memória usando **POKE** são utilizados pela seguinte rotina em Assembly:

```
 10 REM org 58104
 20 REM call cl
 30 REM ld ix,57480
 40 REM ld hl,32
 50 REM ld a,7
 60 REM ld b,255
 70 REM call me
 80 REM ld b,138
 90 REM call me
100 REM ld de,39
110 REM add hl,de
120 REM ld b,100
130 REM ld a,70
140 REM call me
150 REM ktt ld a,253
160 REM in a,254
170 REM bit 1,a
180 REM jr nz,ktt
190 REM ret
200 REM org 58192
210 REM cl *
220 REM org 58155
230 REM me *
```

Esses dois programas — ou o seu produto, criado após a execução ou a montagem pelo Assembler — devem ser gravados da mesma maneira que a primeirar parte do videogame *Avalanche* (veja artigo publicado à página 748).

Para que tal rotina funcione adequadamente, essa parte do jogo e a tabela criada pelo primeiro programa BASIC também devem estar na memória. Isso exige que elas sejam gravadas em código; caso contrário, terão de ser montadas novamente.

Para gravar e ler código de máquina em fita cassete — o que inclui os códigos da tabela ASCII —, o usuário do Spectrum tem à sua disposição comandos **SAVE** e **LOAD** especiais.

Para gravar os códigos de uma porção da memória, digite:

```
SAVE "nome" CODE endereço inici
al,nº de bytes
```

Para recuperar os mesmos códigos da fita cassete, use.

```
LOAD "nome" CODE endereço inici
al
```

O **endereço inicial** pode ser omitido se for igual ao do momento da gravação. Os códigos das diversas partes do videogame *Avalanche* também devem ser gravados; caso isso não aconteça, o usuário gastará cada vez mais tempo na montagem, adicionando novas rotinas e tabelas. Os endereços iniciais e finais devem ser anotados. O próprio Assembler dá essa informação durante a montagem.



Também é bom registrar os endereços dos rótulos, para utilizá-los em instruções com asteriscos (cálculos de saltos e desvios).

ESCREVA AS INSTRUÇÕES NA TELA

A rotina **cl** — que está na primeira parte do programa — é chamada para fazer a limpeza da tela. Já a rotina **me** será usada novamente para escrevermos na tela; para isso, temos que colocar valores adequados em certos registros, que controlam as características do texto a ser impresso.

Para começar, alocamos no registro IX o endereço inicial da nova tabela de códigos ASCII criada pelo programa BASIC. A posição da tela em que começaremos a escrever deve ser colocada no par de registros HL; a cor das letras, em A, e o comprimento do texto, em B. Para imprimir os vários segmentos do texto, é preciso chamar diversas vezes a rotina **me**. Antes de cada chamada, porém, devem ser distribuídos os parâmetros relacionados — posição inicial na tela e número e cor das letras — nos registros adequados.

Quando chamarmos **me** pela primeira vez, B estará contendo o valor 255, que corresponde ao comprimento do texto. Na segunda vez, B estará contendo 138. A rotina deve ser chamada duas vezes porque 255 é o maior número que o registro B, de oito bits, pode abrigar. 138 é o restante do texto, que não pode ser impresso de uma só vez por **me**. Não é preciso definir novamente a posição de impressão na tela — HL —, já que a segunda parte dos dados segue imediatamente a primeira e o conteúdo desse par é atualizado pela própria rotina de impressão (veja a rotina **me** na primeira parte do programa).

Quando quisermos imprimir a terceira parte do texto, contudo, precisaremos usar uma nova posição na tela. Isto é feito para criar um espaço entre o texto inicial — que conta os azares de Willie — e as instruções sobre o uso do teclado, que orientam sobre quais teclas apertar para saltar e/ou correr. Existem duas maneiras de criar esse espaço: colocando uma série de códigos de espaço na tabela ASCII, ou dando um novo valor a HL. Procuramos ilustrar as duas técnicas no programa. Se consultarmos a listagem do programa BASIC que criou a tabela de dados, veremos que existem 26 espaços em branco na linha 320 (naturalmente eles foram transferidos para a tabela). Olhando a listagem Assembly, verificaremos que o número 39 foi colocado em DE pelo comando **ld de,39**. Depois disso, o conteúdo de DE foi somado ao de HL. O par HL funciona como um acumulador de dezesseis bits, de forma que o resultado da operação permanece em HL, que é o local apropriado para o endereço de impressão na tela quando chamamos novamente a rotina **me**.

O restante do texto tem cem caracteres de comprimento e é impresso na cor amarelo brilhante — código 70 — e não mais em branco — código de cor 7 —, como as linhas iniciais.

ESPERE PELO TIRO DE LARGADA

As instruções permanecem no vídeo até que pressionemos a tecla S, dando início ao jogo propriamente dito. Para tornar isso possível, usamos uma rotina que detecta mudanças no teclado por meio do comando **in**.

Esse comando permite o que chamamos de "varredura do teclado", de um modo idêntico ao usado no programa *Trace*, do artigo *Rastreamento no Spectrum* (página 381). Naquela ocasião, queríamos detectar as teclas <**BREAK**> e <**SYMBOL SHIFT**>; agora, nossas atenções estão na tecla S. É o valor colocado em A, antes da execução do comando **in**, que determina a porção do teclado a ser verificada. Assim, **ld a,253** faz com que a linha **in a,254** verifique o conjunto de teclas ao qual S pertence.

O valor recebido através da porta 254 é colocado em A. Se o bit 1 desse número estiver "ligado" é porque S foi pressionada. Quando isso acontece, o resultado da operação **bit 1,a** é zero, ativando o indicador de zero.

O processador ficará girando, preso no laço rotulado como **ktt**, até que a tecla S seja pressionada, ativando o indicador de zero e evitando o salto **jr nz, ktt**, já que a condição **nz** — resultado da última operação não é zero — não foi satisfeita. Normalmente, o programa prosseguiria; por enquanto, porém, uma instrução **ret** provocará o retorno ao BASIC, terminando assim esta segunda parte de *Avalanche*.

Note que os rótulos correspondentes a rotinas montadas em outras listagens devem ter seu endereço definido por **org**, seguido do rótulo e de um asterisco. Esse procedimento permite ao Assembler calcular os saltos para tais rótulos (**cl** e **me**, neste caso). O asterisco impede que a posição de memória correspondente seja alterada.

Se chamarmos o programa tal como foi estruturado até aqui, ele escreverá a tela de instruções logo após a impressão da tela do título, e aguardará que pressionemos a tecla S. A instrução **ret** do final do primeiro programa foi apagada pela montagem da segunda parte, de modo a fazer das duas listagens um todo único e coerente.

Assim como no exemplo precedente, utilizaremos aqui um programa escrito em BASIC para colocar — letra por letra — as instruções na memória, criando uma tabela de códigos ASCII. Adicione as próximas linhas ao programa do artigo anterior. Antes de executá-lo, lembre-se de proteger o topo da memória com **CLEAR 200, 17999**.

```
20 FOR I=1 TO 4
70 NEXT A,I
80 DATA "avalanchecriacao: a.do
```

```
e programa: s. kellaway e g. he
dley"
85 DATA "  depois de um pequeno
 passeio    willie descobre que
 cabritos    monteses espalhara
m todo seu     lanche pela encos
ta.            agora ele tem de
 subir ao topo "
90 DATA " da montanha pegar sua
s coisas.  no caminho ele sera
ameacado    por pedras, cobras
e buracos.    para piorar a situ
acao, a mare  esta subindo, pod
endo afoga-lo. para ajudar will
ie leia as      instrucoes e ap
erte 's' para    jogar.  "
100 DATA "m - correrm - saltarm
n - ambos"
```

Para que a tabela sirva aos nossos propósitos, é importante que o número de códigos de caracteres transferidos para a memória seja exato (afinal, eles vão ser lidos pelo programa em código). Como o leitor pode ter problemas para descobrir o número de espaços em cada linha, é conveniente conferi-los: no final da linha 85, aparecem conjuntos de espaços com os seguintes tamanhos: 2, 1, 4, 1, 4, 4 e 12, segundo a ordem. Na linha 90, a seqüência é a seguinte: 1, 2, 4, 3, 2, 1, 6, 4 e 2. Na linha 100, os espaços têm apenas um caractere em branco.

A seguir, apresentamos o programa em código que utiliza a tabela ASCII criada pela listagem anterior. Para montá-lo na memória, recorra ao Assembler de *INPUT*. Antes disso, porém, use os comandos **POKE** que aumentam a memória disponível e proteja o topo da memória com **CLEAR 200,18999**.

```
10  ORG 19054
20  LDX #1024
30  LDY #17058
40  CLRB
50  JSR LPRINT
60  LDB #137
70  JSR LPRINT
80  LEAX 24,X
90  LDB #10
100  JSR LPRINT
110  LEAX 22,X
120  LDB #10
130  JSR LPRINT
140  LEAX 22,X
150  LDB #10
160  JSR LPRINT
170  KEY JSR 41409
180  CMPA #83
190  BNE KEY
200  RTS
210  LPRINT EQU 19174
```

Esses dois programas — ou o seu produto, criado depois de ter sido executado (ou montado) pelo Assembler — devem ser gravados da mesma maneira que a primeira parte do videogame *Avalanche* (os endereços iniciais e finais devem ser calculados e usados no comando de gravação dos códigos). Para que a rotina acima funcione adequadamente, a primeira parte do jogo e a tabela criada pelo BASIC completo devem estar na memória também.

Faça a gravação dos códigos correspondentes às tabelas e aos programas, empregando o comando **CSAVEM**, que tem a seguinte sintaxe:

```
CSAVEM "nome",endereço inicial,
endereço final
```

ESCREVA AS INSTRUÇÕES NA TELA

Esse programa usa a mesma sub-rotina **LPRINT** que a rotina da página do título empregou para escrever na tela. Note que a tabela ASCII criada pelo programa BASIC é uma continuação da que continha os códigos da página do título, de forma que o programa está utilizando a mesma tabela ASCII.

Toda vez que chamarmos **LPRINT**, devemos colocar nos registros apropriados os parâmetros do texto a ser impresso. Assim, X deve conter a posição da tela onde desejamos iniciar a impressão, B, o comprimento do texto, e Y, a posição do início do texto na tabela.

Se olharmos para a listagem Assembly da sub-rotina **LPRINT** — apresentada na primeira parte —, veremos que o registro Y tem seu valor aumentado na medida em que a tabela ASCII vai sendo "lida" pelo programa. Desse modo, o programa principal não precisará corrigir o valor de Y toda vez que a rotina for chamada; na verdade, Y é aumentado, gradativamente, indicando sempre, linha após linha, o início da porção apropriada da tabela.

Definida a origem do programa, o endereço inicial da tela é colocado em X por **LDX #1024** e o endereço inicial do texto das instruções é posicionado em Y por **LDY #17060**. A instrução **CLRB** limpa o conteúdo de B, sendo um modo rápido de colocar zero nesse registro. O valor zero funciona como se fosse 256; assim, quando a rotina **LPRINT** for chamada por **JSR LPRINT**, serão impressos 256 caracteres.

Ao chamarmos **LPRINT** pela primeira vez, B estará contendo o valor 256, que corresponde ao comprimento do texto. Na segunda vez, B estará contendo 137. A rotina será chamada duas vezes, pois 256 é o maior número que o registro B, de oito bits, pode conter. 138 é o resto do texto que não pode ser impresso todo de uma vez por **LPRINT**.

As instruções **LEAX** somam valores adequados ao conteúdo de X para que a posição de impressão na tela corresponda ao início da linha seguinte; é por isso que não há espaços entre as palavras que ficam no final e no início de linhas adjacentes. A sub-rotina **LPRINT** é chamada então mais quatro vezes para escrever as linhas que orientam as funções das teclas. A cada chamada, o valor de B é acertado por uma instrução **LDB**; o valor de X é ajustado por um **LEAX** (Y é acertado pela própria rotina **LPRINT**).

A sub-rotina **KEY** aguarda que a tecla S seja acionada para que o jogo tenha início. Para verificar o teclado, o programa utiliza uma sub-rotina da ROM. Assim, a instrução **JSR 41409** salta para essa sub-rotina, que verifica qual tecla está sendo pressionada.

Se alguma tecla estiver sendo pressionada, o processador retornará da sub-rotina trazendo o código da tecla correspondente dentro do acumulador. A instrução **CMPA #83** compara então o código da letra S — 83 — com o conteúdo do registro A. Se S não tiver sido acionada, a condição de **BNE KEY** ("salta, se não for igual") estará satisfeita e o microprocessador será enviado novamente ao rótulo **KEY**. O processo se repetirá até que seja pressionada a tecla S. Nesse caso, a condição de **BNE** não estará satisfeita e o programa seguirá seu curso até ser interrompido por uma **RET**. O rótulo correspondente a **LPRINT** deve ter o endereço especificado por uma instrução **EQU**; assim, o Assembler poderá calcular os saltos para aquela sub-rotina.



O Assembly listado a seguir cria a tela de instruções ao jogador. Para fazer isso, ele utiliza dados de uma tabela de códigos ASCII criada por um programa BASIC; nele incluímos uma rotina de "varredura" do teclado, que aguarda que pressionemos a tecla S:

```
10    org -12233
20    ld de,0
30    ld hl,-13958
40    ld bc,960
50    call 92
60    ktt call 159
70    cp 83
80    jr nz,ktt
90    ret
100    end
```

Para montar esse programa, utilize nosso Assembler. Grave o programa-fonte e o programa-objeto da mesma forma que na primeira parte do videogame.

Como das vezes anteriores, precisaremos de um BASIC para colocar na memória os códigos das letras que compõem a tela de instruções. Digite-o, mas não o execute ainda.

```
10 SCREEN 0:CLEAR 200, 14000:KE
Y OFF
20 FOR I=1 TO 22: READ A$: PRIN
T TAB(1),A$: NEXT
30 FOR I=0 TO 959
40 POKE -13958+I,VPEEK (BASE(0)
+I)
50 NEXT
60 CLS
70 DEFUSR=-12288
80 A=USR(0)
90 CLS
100 END
200 DATA "    Após um pequeno p
asseio, Willie"
210 DATA "volta ao local onde p
retendia fazer"
220 DATA "seu pique nique e des
cobre que um"
230 DATA "bando de cabritos mon
teses espalhou"
240 DATA "todo seu lanche na en
costa."
245 DATA
250 DATA "    Agora ele deve su
bir ao topo da "
260 DATA "montanha para recuper
ar suas coisas, "
270 DATA "enfrentando uma avala
nche, cobras"
280 DATA "venenosas e correndo
o risco de cair"
290 DATA "num buraco."
295 DATA
300 DATA "    Para piorar a sit
uação, a maré"
310 DATA "está subindo e ele po
de se afogar."
320 DATA "Se você quer ajudar W
illie, leia as"
330 DATA "instruções e pression
e a tecla 'S'."
335 DATA:DATA
340 DATA "Use:"
350 DATA "    N        para
Correr"
360 DATA "    M        para
Saltar"
370 DATA "    Ambos    para um
Salto Diagonal"
```

As linhas 70 e 80 tentam executar o programa do jogo como foi publicado até o momento. Se a primeira parte dele não estiver na memória ainda, a execução do comando da linha 80 vai apagar toda a memória.

Esse programa utiliza o comando **READ** para obter as frases da tela de instrução nas linhas **DATA** do final da listagem. Depois de escrita, a tela é totalmente transferida para a memória pelo laço entre as linhas 30 e 50. A tabela ASCII é criada assim: o comando **VPEEK** obtém os códigos diretamente na memória de vídeo e o comando **POKE** os coloca na RAM.

Para que a tela mantenha a tabulação original é bom conferir: no início das linhas 200, 250 e 300, há quatro espaços em branco. Nas linhas 350 e 360, cada vazio no texto contém cinco espaços. Uma vantagem dessa técnica de transferência da tela para a memória é que você pode criar sua própria tela de instruções se não estiver satisfeito com o aspecto da nossa.

COMO MONTAR O PROGRAMA INTEIRO

Eis algumas explicações adicionais sobre como montar o programa.

A primeira coisa a fazer é carregar o Assembler na memória. Depois de tomar as providências necessárias para proteger a memória na linha 5000, monte o programa-fonte do artigo anterior. Durante a montagem é interessante anotar o endereço inicial, o endereço final e os endereços dos rótulos.

Terminada a montagem, a rotina em código estará na memória. Quem ainda não a gravou em fita deve fazê-lo por intermédio do comando:

```
BSAVE "CAS:AVAL1",-12288,-12233
```

A seguir, o Assembly deve ser digitado, gravado e montado (anote os endereços principais durante o processo de montagem). A rotina em código pode então ser gravada da mesma maneira.

Depois, digite **NEW**, apagando o Assembler da memória, e carregue o BASIC que cria a tabela ASCII da tela-título, listado no artigo anterior. Quem já gravou a tabela correspondente com o comando **BSAVE** pode carregá-la da fita com **BLOAD**, mas terá que proteger o topo da memória antes. Quem não fez isso ainda deve executar o programa. As linhas finais deste último testam a rotina em código; com a sua execução, a página-título surge na tela; após uma pequena pausa, esta se enche de caracteres aleatórios. Isso acontece porque a rotina tenta escrever a tela de instruções, mas a tabela necessária não se encontra ainda na memória. Para continuar, aperte a tecla S e limpe a tela.

Finalmente, carregue o programa BASIC apresentado linhas atrás e execute-o. As linhas finais testam a rotina completa, que escreve o título, provoca uma pausa, apaga a tela, escreve as instruções e aguarda que a tecla S seja pressionada. A tabela ASCII completa pode ser então gravada com:

```
BSAVE "CAS:ASCII",-14000,-12998
```

É aconselhável gravar os códigos das rotinas e tabelas usando o comando BSAVE. Senão, teremos que gastar um tempo cada vez maior para a montagem. Após gravar os códigos, proteja a memória, digitando a instrução:

```
CLEAR 200,-14000
```

(proteção feita antes pelos programas BASIC). Posicione a fita nas rotinas em código e carregue-as com:

```
BLOAD "CAS:"
```

Finalmente, posicione a fita nos códigos da tabela ASCII e carregue-a usando o mesmo comando **BLOAD**. Para executar o programa, digite:

```
DEFUSR=-12288:A = USR(0)
```

ESCREVA NA TELA

Esse programa recorre à mesma sub-rotina da ROM utilizada pela rotina da página do título para escrever na tela. A tabela ASCII criada pelo programa BASIC é uma continuação da que continha os códigos da página do título, de forma que o programa está utilizando a mesma tabela ASCII.

Toda vez que chamarmos essa sub-rotina, devemos colocar nos registros apropriados os parâmetros do texto a ser impresso. Assim, DE deve conter a posição da tela onde desejamos iniciar a impressão do texto; BC, o comprimento do texto; e HL, a posição do início do texto na tabela ASCII.

A sub-rotina **ktt** aguarda que a tecla S seja pressionada para que o jogo comece. O programa verifica o teclado utilizando uma sub-rotina da **ROM**. Assim, a instrução **call 159** salta para essa sub-rotina, que espera até uma tecla ser pressionada.

Se alguma tecla for pressionada, o processador retornará da sub-rotina trazendo o código da tecla correspondente dentro do acumulador. A instrução **cp 83** compara então o código da letra 2 — 83 — com o conteúdo de A. Se S não tiver sido pressionada, a condição do comando **jr nz,ktt** ("salta, se o último resultado não for zero") estará satisfeita e o microprocessador será novamente enviado para o rótulo **ktt**. O processo se repetirá até que seja pressionada a tecla S. Nesse caso, a condição de **nz** não será satisfeita e o programa prosseguirá seu curso, até encontrar uma instrução **ret**.

Ao executarmos o programa, veremos aparecer um cursor no canto superior esquerdo da tela de instruções. Ele é colocado ali pela rotina da ROM que verifica o teclado. Para tirá-lo dessa posição, teríamos que escrever nossa própria rotina para detectar o toque na tecla S. Depois que a tecla for pressionada e o programa terminar, o texto permanecerá na tela. Sabemos que a rotina terminou porque surge a mensagem "OK" no meio do texto.

TUDO QUE SOBE, DESCE...

Um objeto lançado para o alto descreve uma curva parabólica. Isaac Newton (1642-1727) demonstrou que, se a altura de lançamento for muito grande, o objeto cairá fora da Terra.

Segundo Galileu Galilei (1564-1642), todo objeto lançado sobre a superfície da Terra sofre a ação de duas forças: a que o impulsiona para a frente (ou para o alto) e a que o atrai para o planeta (força de gravidade). A interação dessas duas forças leva o corpo a descrever uma curva parabólica. O desenho desse tipo de trajetória torna-se muito simples quando se trabalha com um computador.

Um objeto lançado para cima, por exemplo, distancia-se — a princípio, rapidamente — das superfície da Terra sob a ação do impulso inicial, perdendo velocidade por efeito da gravidade. Este artigo mostra como programar e simular o movimento de projéteis.

Muitos jogos de combate são ambientados no vácuo do espaço sideral porque lá se pode ignorar a ação da força gravitacional e da resistência do ar no cálculo do movimento de naves e mísseis. Isso não quer dizer que não exista nenhuma gravidade no espaço; existe, mas é tão pequena que pode ser desprezada para efeitos práticos.

Em contrapartida, batalhas simuladas na superfície terrestre têm que levar em conta a ação da gravidade e, freqüentemente, a resistência do ar e o efeito dos ventos. Além de jogos de guerra, podem surgir várias outras circunstâncias em que o conhecimento de como se dá o movimento de projéteis seja essencial para um efeito realístico. Entre elas estão muitos esportes e simulação de provas de atletismo, como lançamento de dardo e de disco, mergulho e todo o tipo de salto.

Os projéteis citados (sejam pessoas ou objetos) têm uma coisa em comum: eles se movem segundo uma trajetória que pertence a um grupo de curvas conhecidas como parábolas. Escrever um programa para simular movimento em trajetória parabólica não é difícil, requerendo apenas a compreensão das forças que atuam sobre o objeto e o emprego de matemática elementar.

Saber, por exemplo, que o movimento executado por uma bola ao ser chutada resulta da combinação de movimento em duas direções já fornece subsídios suficientes para resolver o problema da programação. Uma dessas direções é paralela ao eixo horizontal ou eixo X de um sistema cartesiano. A outra é vertical, paralela ao eixo Y. Nesta lição, estabeleceremos como premissa que o objeto se move com velocidade constante na direção horizontal. Na realidade, o objeto seria desacelerado pela resistência do ar, mas este é um detalhe que costuma ser ignorado, a não ser em trabalhos de grande precisão.

MOVIMENTO HORIZONTAL

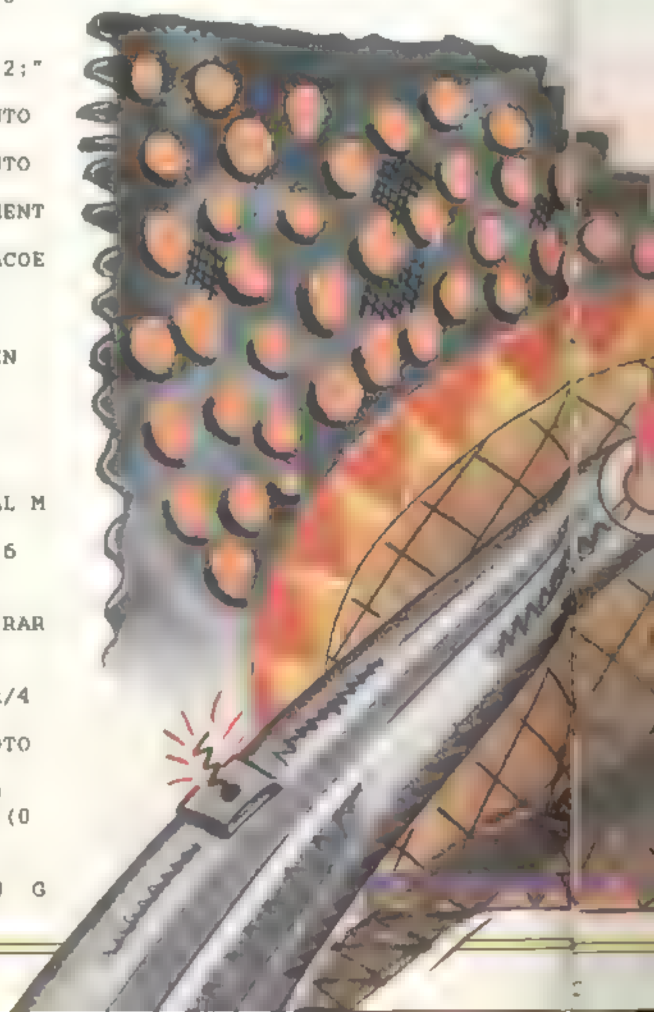
Digite esse primeiro programa para simular o movimento horizontal, e cuidado para não confundir a letra I com o número 1. Todos os programas são para o BASIC padrão da máquina.

S

```
100 BORDER 7· PAPER 7· INK 0·
CLS
105 POKE 23658,8
110 PRINT  INVERSE 1;AT 2,12;"
 MENU "
120 PRINT AT 6,0;"1-MOVIMENTO
PURAMENTE HORIZONTAL"
130 PRINT AT 8,0;"2-MOVIMENTO
PURAMENTE VERTICAL"
140 PRINT AT 10.0;"3- MOVIMENT
O MISTO"
150 PRINT AT 12,0;"4- ELEVACOE
S"
200 LET I$=INKEY$· IF I$=""
THEN  GOTO 200
205 IF I$<"1" OR I$>"4" THEN
 GOTO 200
210 GOSUB VAL I$*1000
220 RUN
1000 CLS
1020 LET SP=30
1030 PRINT "VELOC  HORIZONTAL M
/S..."
1040 FOR R=124 TO 28 STEP -16
1045 LET T=0
1050 PRINT AT 21-(R/8),0;SP
1060 INPUT "<ENTER> PARA ATIRAR
", LINE Z$
1090 LET X=SP*T: LET T=T+1
1100 PLOT 30+X,R: SOUND .1,R/4
1110 PAUSE 10
1120 IF 30+SP*T<250 THEN  GOTO
1090
1150 IF R=28 THEN  GOTO 1200
1160 INPUT "NOVA VELOCIDADE (0
SAI)", LINE I$
1165 LET SP=VAL I$
1170 IF SP<0 OR SP>1000 THEN  G
OTO 1160
1190 IF SP=0 THEN  LET R=28
1200 NEXT R
1210 RETURN
```

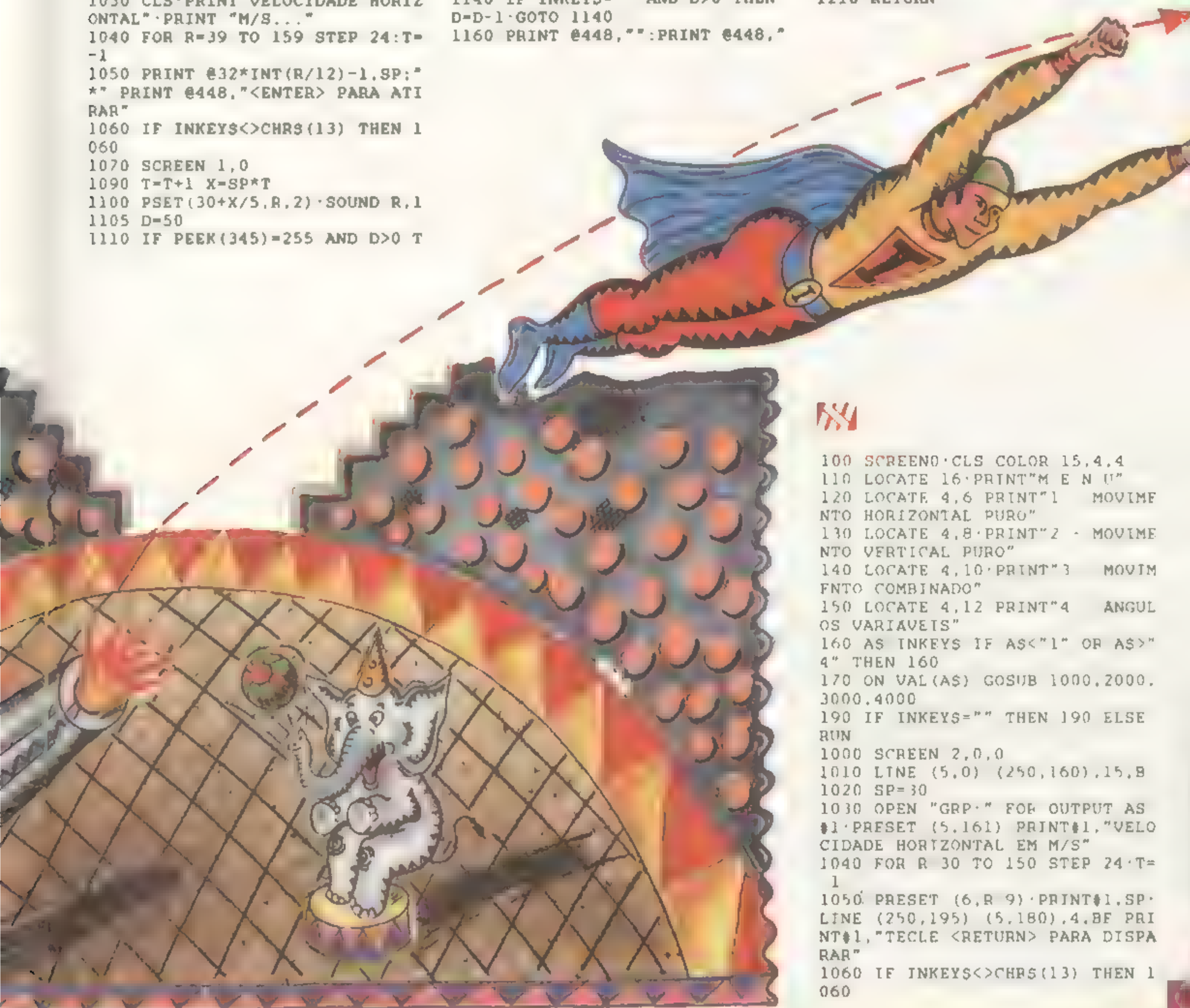


```
100 CLS:PMODE 3
110 PRINT @13,"menu"
120 PRINT @128,"1-MOVIMENTO PUR
AMENTE HORIZONTAL"
130 PRINT @192,"2-MOVIMENTO PUR
AMENTE VERTICAL"
140 PRINT @256,"3-MOVIMENTO MIS
TO"
150 PRINT @320,"4-ELEVACOES"
160 A$=INKEY$:IF A$<"1" OR A$>"
4" THEN 160
170 ON VAL(A$) GOSUB 1000,2000,
3000,4000
```



- COMO DESENHAR A TRAJETÓRIA DE PROJÉTEIS
- * EFEITOS DE DIFERENTES CAMPOS GRAVITACIONAIS
- COMO SIMULAR PARÁBOLAS
- FUNCIONAMENTO DO PROGRAMA
- MOVIMENTO VERTICAL
- MOVIMENTO HORIZONTAL
- COMO UNIR AS ROTINAS
- MUDE O ÂNGULO

```
180 AS=INKEYS
190 IF INKEYS="" AND PEEK(65314
)<>7 THEN 190 ELSE RUN
1000 PCLS
1010 LINE(0,0)-(255,191),PSET,B
1020 SP=30
1030 CLS·PRINT"VELOCIDADE HORIZ
ONTAL"·PRINT "M/S..."
1040 FOR R=39 TO 159 STEP 24:T=
-1
1050 PRINT @32*INT(R/12)-1,SP;"
*" PRINT @448,"<ENTER> PARA ATI
RAR"
1060 IF INKEYS<>CHRS(13) THEN 1
060
1070 SCREEN 1,0
1090 T=T+1 X=SP*T
1100 PSET(30+X/5,R,2)·SOUND R,1
1105 D=50
1110 IF PEEK(345)=255 AND D>0 T
HEN D=D-1·GOTO 1110
1120 IF SP*(T+1)/5<223 THEN 109
0
1130 IF R>150 THEN 1200
1135 D=200
1139 AS=INKEY$
1140 IF INKEYS="" AND D>0 THEN
D=D-1·GOTO 1140
1160 PRINT @448,"":PRINT @448,"
NOVA VELOCIDADE (0 SAI):";:INPU
T SP
1170 IF SP<0 OR SP>999 THEN 116
0
1190 IF SP=0 THEN R=160
1200 NEXT
1210 RETURN
```



```
100 SCREEN0·CLS COLOR 15,4,4
110 LOCATE 16·PRINT"M E N U"
120 LOCATE 4,6 PRINT"1   MOVIME
NTO HORIZONTAL PURO"
130 LOCATE 4,8·PRINT"2 - MOVIME
NTO VERTICAL PURO"
140 LOCATE 4,10·PRINT"3   MOVIM
ENTO COMBINADO"
150 LOCATE 4,12 PRINT"4   ANGUL
OS VARIAVEIS"
160 AS INKEYS IF AS<"1" OR A$>"
4" THEN 160
170 ON VAL(A$) GOSUB 1000,2000,
3000,4000
190 IF INKEYS="" THEN 190 ELSE
RUN
1000 SCREEN 2,0,0
1010 LINE (5,0) (250,160),15,B
1020 SP=30
1030 OPEN "GRP·" FOR OUTPUT AS
#1·PRESET (5,161) PRINT#1,"VELO
CIDADE HORIZONTAL EM M/S"
1040 FOR R 30 TO 150 STEP 24·T=
 1
1050 PRESET (6,R 9)·PRINT#1,SP·
LINE (250,195) (5,180),4,BF PRI
NT#1,"TECLE <RETURN> PARA DISPA
RAR"
1060 IF INKEYS<>CHRS(13) THEN 1
060
```

```
1090 T=T+1·X=SP*T
1100 PSET (30+X/5,R),15 PLAY"N1
9L19"
1110 FOR I=1 TO 100·IF INKEY$ <
>CHR$(32) THEN NEXT I
1120 IF SP*(T+1)/5+30<240 THEN
1090
1130 IF R>145 THEN 1200
1140 FOR I=1 TO 200 IF INKEY$<>
CHR$(13) THEN NEXT I
1160 LINE (250,195) (5,180),4,B
F·PRINT#1,"NOVA VELOC? (0 TERMI
NA) " SP$=""
1162 PRESET (210,180)·PRINT#1,S
P$
1165 A$ INKEY$·IF A$=CHR$(13) T
HEN 1190
1170 IF A$<"0" OR A$>"9" THEN 1
165
1180 SP$=SP$+A$·GOTO 1162
1190 SP=VAL(SP$) IF SP=0 THEN R
=160
1195 IF SP<0 OR SP>999 THEN 116
0
1200 NEXT R
1210 RETURN
```



```
95  FOR I = 770 TO 795· READ A·
 POKE I,A· NEXT
100  TEXT   HOME
110  PRINT  TAB( 16);"M E N U"
120  VTAB 7: HTAB 5: PRINT "1 -
 MOVIMENTO HORIZONTAL PURO"
130  VTAB 9: HTAB 5: PRINT "2 -
 MOVIMENTO VERTICAL PURO"
140  VTAB 11  HTAB 5  PRINT "3
- MOVIMENTO COMBINADO"
150  VTAB 13  HTAB 5· PRINT "4
- ANGULOS VARIAVEIS"
160  GET A$· IF A$ < "1" OR A$
> "4" THEN 160
170  ON  VAL (A$) GOSUB 1000,20
00,3000,4000
180  GET A$: IF A$ =  CHR$ (13)
 THEN  RUN
190  GOTO 180
1000  HGR : HCOLOR= 3
1010  HPLOT 0,0 TO 279,0 TO 279
,159 TO 0,159 TO 0,0
1020 SP = 30
1030  HOME
1040  FOR R = 30 TO 150 STEP 24
:T =  - 1
1050  VTAB 21  PRINT "VELOC HOR
IZ: ";SP;" M/S": VTAB 24: PRINT
 "TECLE <CR> PARA DISPARAR ".
1060  GET A$  IF A$ <  >  CHR$
(13) THEN 1060
1090 T = T + 1·X = SP * T
1100  HPLOT X / 5,R:W =  PEEK (
 - 16336):W =  PEEK ( - 16336)
1105  POKE  - 16368,0
1110  FOR D = 1 TO 20: IF  PEEK
 ( - 16384) < 128 THEN  POKE  -
 16368,0: NEXT
```

```
1120 IF SP * (T + 1) / 5 < 279
 THEN 1090
1130 IF R > 149 THEN 1200
1135 POKE   16368,0
1140 FOR D =   TO 300: IF  PEE
K ( - 16384) > 127 THEN D = 300

1150 POKE  - 16368,0: NEXT
1160 HTAB 1: VTAB 23: CALL
958  VTAB 23  INPUT "NOVA VELOC
IDADE (0 TERMINA  ",SP
1170 IF SP < 0 OR SP > 999 THE
N 1160
1190 IF SP = 0 THEN R = 160
1200 NEXT
1210 RETURN
5000 DATA  172,1,3,174,1,3,169
,4,32,168,252,173,48,192,232,20
8,253,136,208,239,206,0,3,208,2
31,96
```

Ao executar o programa, você verá um menu com quatro opções. Até agora, só a rotina para a primeira opção foi digitada. Assim, se teclar as opções 2, 3 ou 4, você obterá uma mensagem de erro. Ao pressionar 1, o programa vai para a sub-rotina 1000. Esta usa um laço **FOR...NEXT** (linhas 1040 a 1200) para permitir que um objeto seja lançado horizontalmente em seis velocidades diferentes. Na primeira vez, a velocidade de 30 m por segundo é usada automaticamente e simulada no vídeo como uma série de pontos em linha reta.

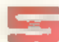
A seguir, você pode escolher uma velocidade diferente. Para sair do laço, pressione a tecla 0; o programa mostrará então o menu novamente. Digite um valor — 60, por exemplo — e pressione **<ENTER>** ou **<RETURN>** para disparar. A ação pode ser acelerada acionando-se continuamente a barra de espaços (tecla **<ENTER>** no Spectrum) ou teclando-a repetidas vezes. Agora, compare o resultado com a linha de velocidade 30 m/s. Escolha e compare cinco outras velocidades. Depois disso, o menu será reapresentado.

A parte do programa que desenha os pontos fica entre as linhas 1090 e 1120. A variável **T** simula o tempo que, neste caso, é incrementado de um em um segundo. Desse modo, os pontos são espaçados regularmente (o que é necessário para conservar a velocidade constante) na direção horizontal. A distância entre os pontos é determinada pela expressão **SP*T** na linha 1090. Ela faz com que, quanto maior a velocidade, maior o espaçamento entre os pontos.

A expressão **SP*T** pode ser reconhecida como parte da fórmula DISTÂNCIA = VELOCIDADE X TEMPO, como é ensinada nas aulas de física. Desse modo, fica bem claro que o programa usa o espaçamento entre os pontos, a fim de simular a velocidade.

MOVIMENTO VERTICAL

Digite as próximas linhas para simular o movimento em direção vertical:



```
2000 CLS
2020 LET G=10  LET SP=50
2030 PRINT '"ACELERACAO DA GRAV
 : m/s/s  ."
2040 FOR R=3 TO 18 STEP 3
2050 PRINT AT 20,R;"*";AT 21,R;
G
2060 INPUT "<ENTER> PARA ATIRAR
", LINE I$
2080 FOR T=0 TO 250 STEP  5
2090 LET H=SP*T 0 5*G*T*T
2100 IF H>143 THEN  SOUND  05,1
0: PAUSE 50  NEXT T  GOTO 2110
2105 IF H>=0 THEN  PLOT R*8+4,H
+32: SOUND .05,H/4: PAUSE 40: N
EXT T
2110 IF R>15 THEN  GOTO 2180
2140 INPUT "NOVA GRAV. (0 SAI)"
, LINE I$
2142 IF LEN I$=0 THEN  GOTO 214
0
2145 LET G=VAL I$
2150 IF NOT (LEN I$>0 AND G>=0
AND G<400) THEN  GOTO 2140
2170 IF G=0 THEN  LET R=18
2180 NEXT R
2190 RETURN
```



```
2000 PCLS
2010 LINE(0,0)-(255,191),PSET,B
2020 G=10:SP=50:CLS
2030 PRINT @32,"ACELERACAO DA G
RAV.:":PRINT"M/S/S..."
2040 FOR R=0 TO 25 STEP 5
2050 PRINT @416+R,"*"+MID$(STR$
(G),2);
2060 PRINT @448,"<ENTER> PARA A
TIRAR"
2065 IF INKEY$<>CHR$(13) THEN 2
065
2070 SCREEN 1,0
2080 FOR T=0 TO 100 STEP .5
2090 H=SP*T-.5*G*T*T
2095 IF H<0 THEN T=250:GOTO 210
8
2100 IF H>159 THEN SOUND 250,1:
D=30 ELSE PSET (10+R*8,160-H,2)
:SOUNDH+10,1:D=50
2105 IF PEEK(345)=255 AND D>0 T
HEN D=D-1:GOTO 2105
2108 NEXT
2110 IF R>20 THEN 2180
2115 D=80
2119 A$=INKEY$
2120 IF INKEY$="" AND D>0 THEN
D=D-1:GOTO 2120
2140 PRINT @448,"":PRINT @448,"
NOVA GRAV.(0 SAI):";:INPUT G
2150 IF G<0 OR G>400 THEN 2140
2170 IF G=0 THEN R=26
2180 NEXT
2190 RETURN
```



```
2000 SCREEN 2,0,0
2010 LINE (5,0)-(250,160),15,B
2020 SP=50:G=10
2030 OPEN "GRP:" FOR OUTPUT AS
#1 PRESET (5,161) PRINT#1,"ACEL
 DA GRAVIDADE EM M/S/S"
2040 FOR R=0 TO 25 STEP 5
2050 PRESET (R*8,150) PRINT#1,G
 LINE (250,195) 5,180),4,BF PR
INT#1,"TECLE <RETURN> PARA DISP
ARAR"
2060 IF INKEY$<>CHR$(13) THEN 2
060
2080 FOR T=0 TO 100 STEP  5
2090 H=SP*T-.5*G*T*T
2095 IF H<0 THEN T=250:GOTO 210
8
2100 IF H>159 THEN PLAY"N90L19"
 ELSE PSET (30+R*8,160-H),15:A$
 "N"+STR$(INT(H/2))+"L19" PLAYA
$
2105 FOR I=1 TO 100 IF INKEY$ <
>CHR$(32) THEN NEXT I
2108 NEXT T
2110 IF R 20 THEN 2180
2120 FOR I=1 TO 200:IF INKEY$ <
 CHR$(32) THEN NEXT I
2140 LINE (250,195)-(5,180),4,B
F:PRINT#1,"NOVA G? (0 TERMINA):
" G$=""
2142 PRESET (200,180) PRINT#1,G
$
2145 A$=INKEY$:IF A$=CHR$(13) T
HEN 2160LIST 2100-
2150 IF A$<"0" OR A$>"9" THEN 2
145
2155 G$=G$+A$ GOTO 2142
2160 G=VAL(G$) IF G 0 THEN R=26
2170 IF G<0 OR G>400 THEN 2140
2180 NEXT R
2190 RETURN
```



```
2000  HGR : HCOLOR= 3
2010  HPLOT 0,0 TO 279,0 TO 279
,159 TO 0,159 TO 0,0
2020 SP = 50:G = 10
2030  HOME : VTAB 21  PRINT "G
 M/S/S"
2040  FOR R = 0 TO 25 STEP 5
2050  VTAB 21: HTAB R + 8: PRIN
T G: VTAB 24: PRINT "TECLE <CR>
 PARA DISPARAR ";
2070  GET A$: IF A$ <  >  CHR$
(13) THEN 2070
2080  FOR T = 0 TO 100 STEP .5
2090 H = SP * T - .5 * G * T *
T
2095  IF H < 0 THEN T = 250: GO
TO 2107
2100  IF H > 159 THEN  POKE 769
,200: POKE 768,2: CALL 770
2102  IF H < 160 THEN  HPLOT 50
 + R * 7,160 - H: POKE 769,H: P
OKE 768,2: CALL 770
2103  POKE  - 16368,0
2105  FOR D = 1 TO 20: IF  PEEK
 ( - 16384) < 128 THEN  POKE  -
 16368,0: NEXT
2107  NEXT T
```

```
2110  IF R > 20 THEN 2180
2115  POKE  - 16368,0
2120  FOR D = 1 TO 300: IF  PEE
K ( - 16384) > 127 THEN D = 300

2130  POKE  - 16368,0: NEXT
2140  HTAB 1: VTAB 23: CALL  -
958: VTAB 23: INPUT "NOVA ACELE
RACAO (0 TERMINA) ";G
2150  IF G < 0 OR G > 400 THEN
2140
2170  IF G = 0 THEN R = 26
2180  NEXT R
2190  RETURN
```

Execute o programa e escolha a opção 2 desta vez. Pressione <ENTER> ou <RETURN> para ver uma série de pontos traçados verticalmente na tela. Note que tais pontos não são espaçados regularmente, como na opção anterior; quanto mais alta sua localização, mais perto eles ficam uns dos outros.

Isso acontece porque o objeto é desacelerado pela ação da gravidade. E desta vez o som ajuda a explicar o que está acontecendo. À medida que o objeto sobe, a tonalidade do som torna-se mais aguda e, à proporção que ele desce, o som fica mais grave.

Como antes, a rotina lhe oferece seis tentativas (determinadas na linha 2040) de experimentar diferentes valores para o efeito da gravidade. Elas são armazenadas na variável **G**, que inicialmente é ajustada para 10 (linha 2020) e usada na linha 2090.

A relação é a fórmula para distância de um objeto em queda livre. Sua forma usual é:

$$S = VT + \frac{1}{2}GT^2,$$

onde S é a distância; V, a velocidade inicial; T, o tempo, e G, a aceleração da gravidade. TxT é usado na linha 2090 em vez de T↑2 ou T2, porque os computadores são mais ágeis para multiplicar do que para calcular potências. A aceleração de um objeto em movimento é calculada em função da mudança de velocidade e do tempo decorrido. Perto da superfície da Terra, g tem um valor aproximado 10 m/s/s. Isso significa que a velocidade de um objeto em queda aumenta de 10 m/s a cada segundo. E a velocidade de um objeto em ascensão diminui de 10 m/s a cada segundo. Esta é, portanto, uma aceleração negativa; por isso, o sinal + (mais) da fórmula padrão é substituída pelo − (menos) na linha 2090.

O cálculo de **G** é comumente usado em relação a viagens espaciais. A aceleração de uma espaçonave saindo da órbita terrestre atinge cerca de 10 g. Isso significa que essa aceleração é de 10x10 m/s/s ou 100 m/s/s (a notação m/s também é usada com freqüência).

Na primeira passada do laço que começa na linha 2040, a posição do objeto é traçada a intervalos de um segundo. A velocidade inicial é de 50 m/s e a aceleração é a da gravidade (10 m/s/s). Ambos os valores foram determinados na linha 2020. Como na primeira rotina, apressa-se a ação pressionando-se a barra de espaços (ou **<ENTER>**, no Spectrum). O valor de **G** pode ser mudado quando um novo valor é solicitado na tela. (Compare o efeito dos seis valores diferentes.) Antes de se completar o laço, pode-se voltar ao menu pressionando-se 0 para aceleração.

MOVIMENTO COMBINADO

Para simular o movimento de um projétil, é necessário apenas combinar essas rotinas de maneira que o objeto se mova na horizontal e na vertical, ao mesmo tempo. Digite estas linhas para montar a terceira rotina:

```
3000 CLS
3020 LET G=10: LET SP=50
3030 FOR R=1 TO 6
3040 PRINT AT 0,0;"GRAV =";G;"m
/s/s "'"VELOCIDADE=";SP;"m/s "
3050 INPUT "<ENTER> PARA ATIRAR
", LINE I$
3070 FOR T=0 TO 250 STEP .5
3080 LET H=SP*SIN ((PI/180)*45)
*T- 5*G*T*T
3090 LET X=SP*COS ((PI/180)*45)
*T
3100 IF H<175 AND X<255 THEN  G
OTO 3105
3102 IF H>0 THEN  SOUND .05,10:
 PAUSE 25: NEXT T: GOTO 3110
3103 LET T=250: NEXT T: GOTO 31
10
3105 IF H>=0 THEN  PLOT X,H: SO
UND .05,H/4: PAUSE 40: NEXT T:
GOTO 3110
3106 LET T=250: NEXT T
3110 IF R=6 THEN  GOTO 3230
3120 PAUSE 50
3140 INPUT "NOVA GRAV. (0 SAI)"
, LINE I$
3150 IF LEN I$=0 THEN  GOTO 314
0
3155 LET G=VAL I$
3160 IF NOT (LEN I$>0 AND G>=0
AND G<1000) THEN  GOTO 3140
3170 IF G=0 THEN  LET R=6: GOTO
 3230
3190 INPUT "NOVA VELOC.", LINE
I$
3195 LET SP=VAL I$
3200 IF NOT (LEN I$>0 AND SP>0
AND SP<1000) THEN  GOTO 3190
3230 NEXT R
3240 RETURN
```



```
3000 PCLS
3010 LINE(0,0)-(255,191),PSET,B
3020 G=10:SP=50
3030 FOR R=1 TO 6:CLS
3040 PRINT "G     =";G;"M/S/S":P
RINT"VELOCIDADE=";SP;"M/S"
3050 PRINT @448,"<ENTER> PARA A
TIRAR"
3060 IF INKEY$<>CHR$(13) THEN 3
060
3065 SCREEN 1,0
3070 FOR T=0 TO 200 STEP .5
3080 H=SP*SIN(ATN(1))*T-.5*G*T*
T
3090 X=SP*COS(ATN(1))*T
3095 IF H<0 THEN T=250:GOTO 310
6
3100 IF X>251 THEN T=250:GOTO 3
106 ELSE IF H>189 THEN SOUND 25
0,1:D=25 ELSE PSET(X+2,190-H,(R
+3)/2):SOUND H+10,1:D=35
3104 IF PEEK(345)=255 AND D>0 T
HEN D=D-1:GOTO 3104
3106 NEXT
3110 IF R>5 THEN 3230
3115 D=100
3119 A$=INKEY$
3120 IF INKEY$="" AND D>0 THEN
D=D-1:GOTO 3120
3130 PRINT @416,"":PRINT @448,"
"
3140 PRINT @416,"NOVA GRAV. (0
SAI)";:INPUT G
3160 IF G<0 OR G>9999 THEN 3130
3170 IF G=0 THEN R=6:GOTO 3230
3180 PRINT @448,""
3190 PRINT @448,"NOVA VELOC.:";
:INPUT SP
3200 IF SP<0 OR SP>999 THEN 318
0
3230 NEXT
3240 RETURN
```



```
3000 SCREEN 2,0,0
3010 LINE (5,0)-(250,160),15,B
3020 SP 50 G=10
3025 OPEN "GRP:" FOR OUTPUT AS
#1
3030 FOR R=1 TO 6
3040 LINE (250,195) (5,161),4,B
F PRINT#1,"G =";G,"M/S/S      VE
L  ",SP;"M/S"
3050 LINE (250,195) (5,180),4,B
F PRINT#1,"TECLE <RETURN> PARA
DISPARAR"
3060 IF INKEY$< CHR$(13) THEN 3
060
3070 FOR T=0 TO 200 STEP  5
3080 H=SP*SIN(ATN(1))*T  5*G*T*
T
3090 X=SP*COS(ATN(1))*T
3095 IF H<0 THEN T=250 GOTO 310
6
3100 IF X>245 THEN T=250:GOTO 3
106 ELSE IF H>159 THEN PLAY"N90
L19" ELSE PSET (X+6,160-H),15:A
$="N"+STR$(INT(H/2))+"L19" PLAY
```

```
A$
3104 FOR I=1 TO 100: IF INKEY$ <
CHR$(13) THEN NEXT I
3106 NEXT T
3110 IF R=5 THEN 3230
3120 FOR I=1 TO 200: IF INKEY$
CHR$(32) THEN NEXT I
3140 LINE (250,195)-(5,180),4,B
F: PRINT#1,"NOVA G? (0 TERMINA)
": G$=""
3142 PRESET (200,180): PRINT#1,G
$
3145 A$=INKEY$:IF A$=CHR$(13) T
HEN 3160
3150 IF A$<"0" OR A$>"9" THEN 3
145
3155 G$=G$+A$: GOTO 3142
3160 G=VAL(G$): IF G=0 THEN R=6:
GOTO 3230
3170 IF G<0 OR G>9999 THEN 3140
3180 LINE (250,195)-(5,180),4,B
F:PRINT#1,"NOVA VELOC?":SP$=""
3182 PRESET (200,180): PRINT#1,S
P$
3185 A$=INKEY$:IF A$=CHR$(13) T
HEN 3210
3190 IF A$<"0" OR A$>"9" THEN 3
185
3195 SP$=SP$+A$: GOTO 3182
3210 SP=VAL(SP$)
3220 IF SP<0 OR SP>999 THEN 318
0
3230 NEXT R
3240 RETURN
```



```
3000  HGR : HCOLOR= 3
3010  HPLOT 0,0 TO 279,0 TO 279
,159 TO 0,159 TO 0,0
3020 G = 10:SP = 50
3030  FOR R = 1 TO 6: HOME
3040  VTAB 21: PRINT "G  ";G;"
M/S/S   ";"VELOCIDADE  ";SP;" M
/S": VTAB 24: PRINT "TECLE <CR>
 PARA DISPARAR ";
3060  GET A$: IF A$ < > CHR$
(13) THEN 3060
3070  FOR T = 0 TO 200 STEP .5
3080 H = SP *  SIN ( ATN (1)) *
 T - .5 * G * T * T
3090 X = SP *  COS ( ATN (1)) *
 T
3095  IF H < 0 THEN T = 250: GO
TO 3125
3100  IF X > 275 THEN T = 250:
GOTO 3125
3105  IF H > 160 THEN  POKE 769
,200: POKE 768,2: CALL 770
3110  IF H < 160 THEN  HPLOT X
+ 2,160 - H: POKE 769,H: POKE 7
68,2: CALL 770
3115  POKE  - 16368,0
3120  FOR D = 1 TO 20: IF  PEEK
 ( - 16384) < 128 THEN  POKE  -
 16368,0: NEXT
3125  NEXT T
3130  IF R > 5 THEN 3230
3135  POKE  - 16368,0
3140  FOR D = 1 TO 300: IF  PEE
K ( - 16384) > 127 THEN D = 300

3145  POKE  - 16368,0: NEXT
3150  HTAB 1: VTAB 23: CALL
958: VTAB 23: INPUT "NOVA ACELE
RACAO (0 TERMINA) ";G
3160  IF G < 0 OR G > 9999 THEN
 3150
3170  IF G = 0 THEN R = 6: GOTO
 3230
3180  HTAB 1: VTAB 23: CALL
958: VTAB 23: INPUT "NOVA VELOC
IDADE (0 TERMINA) ";SP
3200  IF SP < 0 OR SP > 999 THE
N 3180
3230  NEXT R
3240  RETURN
```

Execute o programa e escolha a opção 3. Após <ENTER> ou <RETURN> terem sido pressionados, serão traçados pontos segundo uma curva que começa no canto inferior esquerdo da tela e termina em algum lugar em direção ao canto inferior direito. Essa é a trajetória de um objeto lançado a uma velocidade inicial de 50 m/s sob gravidade G.

ESTRUTURA DA ROTINA

A estrutura da rotina é similar às anteriores. O cálculo e o traçado são feitos em um laço **FOR...NEXT** (linhas 3030 a 3230), que permite comparar seis trajetórias diferentes, cinco das quais especificadas pelo usuário. Um valor 0 provoca o retorno ao menu. Mas é mais provável que você queira digitar um novo conjunto de valores para comparar as trajetórias.

Coloque o valor 5 em **G** e mantenha **SP** em 50; pressione **<ENTER>**. Desta vez, o objeto irá mais alto e mais longe. Mantenha **G** em 5 e reduza **SP** para 25. Compare os resultados.

Continue experimentando: modifique tanto **G** quanto **SP** e preste atenção aos sons. Estes foram especialmente planejados para ajudá-lo a compreender o movimento do objeto na tela. Dessa maneira, sons cada vez mais agudos indicam um movimento ascendente. A queda, por sua vez, é marcada por sons cada vez mais graves.

COMO FUNCIONA

Lembre-se de que a trajetória do objeto é traçada como coordenadas **H** no eixo Y e coordenadas **X** no eixo X. Essas duas variáveis são calculadas nas linhas 3080 e 3090. A única diferença entre elas é que a coordenada **H** tem a velocidade (**SP**) multiplicada pelo seno do ângulo e a coordenada **X** tem **SP** multiplicada pelo co-seno do mesmo ângulo (45°). Isso explica por que, quando **G** é pequena e **SP** grande, a trajetória parece uma linha diagonal a 45°.

SENO, CO-SENO, TANGENTE

Essas funções trigonométricas são necessárias para calcular a fração do movimento que se aplica a cada uma das duas direções. Tais frações, por sua vez, são chamadas de componentes horizontal e vertical do movimento. Se este tem uma velocidade inicial de 50 m/s, por exemplo, as duas componentes serão menores que 50. Somadas, elas dão exatamente 50 m/s.

Como você deve estar lembrado, a componente vertical é **SP*SEN A** e a horizontal, **SP*COS A**, onde A é o ângulo de inclinação do lançamento.

VISUALIZE O MOVIMENTO

Para compreender como esses valores são obtidos, é conveniente dar uma olhada em um esquema. O desenho da página 768 mostra um projétil iniciando seu movimento com velocidade real **V** em um ângulo **A** com a horizontal. As linhas tracejadas, por sua vez, mostram as componentes da velocidade (**Vh** e **Vx**) nas duas direções.

O seno do ângulo **A** é **Vh/V**; temos assim uma relação que pode ser lida como **Vh = V*SEN A**. Da mesma forma, o co-seno de **A** é **Vx/V**, de onde inferimos a equação **Vx = V*COS A**.

Seguindo esse padrão, temos **SP*SEN 45** para a componente vertical da velocidade e **SP*COS 45** para a componente horizontal.

No programa, isso aparece de um modo um pouco diferente. À exceção do Spectrum, usa-se a função **ATN(1)**. Essa função (arco tangente) fornece o ângulo cuja tangente é 1 (nosso caso): o ângulo de 45°. O computador, porém, utiliza tal valor em radianos. Por isso, recorremos à função — que fornece o valor na unidade adequada — e não ao valor 45, diretamente. No Spectrum usa-se o valor **pi/180**, que equivale a 1 grau, multiplicado por 45.

COMO MUDAR O ÂNGULO

Neste ponto, você pode estar querendo saber por que o ângulo foi fixado em 45°. Na verdade, tanto o ângulo quanto a velocidade inicial podem ser alterados para mudar a trajetória do projétil. Mais frequentemente, varia-se o ângulo, deixando a velocidade constante. É o que faremos agora:

MICRO DICAS

COMO É FEITA A SIMULAÇÃO DE UM MOVIMENTO

Todos os programas que simulam fenômenos físicos de natureza contínua, como o movimento de um corpo no espaço, utilizam o mesmo tipo de modelo matemático: as equações de diferenças finitas. Por esse motivo, é interessante saber qual o seu significado e como funcionam. Assim, seus esforços de programação em muitas áreas, inclusive jogos, serão bem mais profissionais.

Uma equação de diferenças finitas diz qual é a lei matemática que rege a maneira pela qual uma variável muda com o tempo. Tomemos como exemplo o movimento de um corpo com velocidade constante. Suponhamos nesse caso que a distância percorrida pelo corpo em um intervalo de tempo bem pequeno — que chamaremos de **DT** — é **DD**. Como conseqüência, a equação que rege **DD** em função de **DT** é:

$$DD = V^{*}DT,$$

onde V é a velocidade do corpo.

Na realidade, o que acontece na natureza é um movimento contínuo, ou seja, podemos imaginar um **DT** infinitesimalmente pequeno. Porém, se dermos agora o nome **D** à distância total percorrida pelo corpo, teremos a seguinte equação, que é bem mais útil que a primeira:

$$D = D + V^{*}T$$

Essa é a forma a ser colocada em um programa de computador e se chama equação de diferenças finitas. Ela é calculada repetidamente, começando com o valor de **D** = 0. O valor de **D** é então incrementado a cada intervalo **DT** por um "pedacinho" calculado pela fórmula **V*DT**. Esse processo simples corresponde a uma integração da distância ao longo do tempo (integração é um termo específico da parte da matemática chamada cálculo diferencial e integral).

Como não é possível trabalhar em um computador com uma grandeza infinitamente pequena para **DT**, usa-se um **DT** pequeno mas finito.

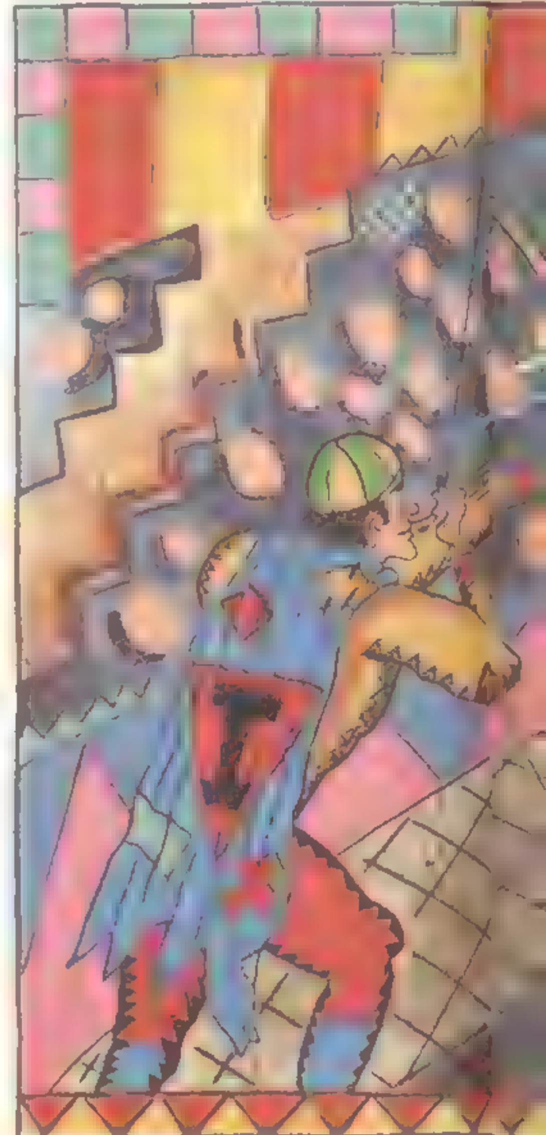
```
4000 CLS
4010 LET FL=0
4020 RESTORE : FOR N=0 TO 7: RE
AD A  POKE USR "A"+N,A· NEXT N
4040 LET A=70· LET SP=50
4060 PRINT AT 0,0;"ANGULO=";A;C
HR$ 144;CHR$ 32
4070 INPUT "<ENTER> PARA ATIRAR
", LINE I$
4080 FOR T=0 TO 250 STEP .5
4090 LET H=SP*SIN ((PI/180)*A)*
T- 5*10*T*T· LET X=50*COS ((PI/
180)*A)*T
4100 IF H>=0 THEN  PLOT X,H+16:
 SOUND .05,H/4: PAUSE 40: NEXT
T: GOTO 4110
4105 LET T=250: NEXT T
4110 PAUSE 50
4130 INPUT "NOVO ANGULO (0 SAI)
", LINE I$
4135 IF LEN I$=0 THEN  GOTO 413
0
4140 LET A=VAL I$
4150 IF NOT (LEN I$>0 AND A>=0
AND A<90) THEN  GOTO 4130
4160 IF A=0 THEN  LET FL=1
4170 IF NOT FL THEN  GOTO 4060
4180 RETURN
5000 DATA 24,36,36,36,24,0,0 0
```



```
4000 PCLS
4020 LINE (0,0)-(255,191),PSET,
B
4040 A=70:SP=50
4060 CLS:PRINT "ANGULO = ";A;"G
RAUS"
4070 PRINT @448,"<ENTER> PARA A
TIRAR"
4072 IF INKEY$<>CHR$(13) THEN 4
072
4075 SCREEN 1,0:AN=A*ATN(1)/45
4080 FOR T=0 TO 250 STEP .5
4090 H=SP*SIN(AN)*T-.5*10*T*T:X
=SP*COS(AN)*T
4092 IF H<0 THEN T=250:GOTO 410
0
4094 IF X>251 THEN T=250:GOTO 4
100 ELSE IF H>189 THEN SOUND 25
0,1:D=25 ELSE PSET(X+2,190-H,2)
:SOUND H+10,1:D=35
4096 IF PEEK(345)=255 AND D>0 T
HEN D=D-1:GOTO 4096
4100 NEXT
4110 A$=INKEY$
4120 IF INKEY$="" THEN 4120
4130 PRINT @448,"NOVO ANGULO (0
 SAI):";:INPUT A
4160 IF A<0 OR A>=90 THEN 4120
4170 IF A>0 THEN 4060
4180 RETURN
```



```
4000 SCREEN 2,0,0
4020 LINE (5,0)-(250,160),15,B
4040 SP=50·A=70
4050 OPEN "GRP·" FOR OUTPUT AS
#1
4060 LINE (250,175)-(5,161),4,B
F·PRINT#1,"ANGULO =";A;"GRAUS"
4070 LINE (250,195)-(5,180),4,B
F·PRINT#1,"TECLE <RETURN> PARA
DISPARAR"
4072 IF INKEY$<>CHR$(13) THEN 4
072
4075 AN=A*ATN(1)/45
4080 FOR T=0 TO 250 STEP .5
4090 H=SP*SIN(AN)*T-.5*10*T*T·X
=SP*COS(AN)*T
4092 IF H<0 THEN T=250·GOTO 410
0
4094 IF X>245 THEN T=250 GOTO 4
100 ELSE IF H>159 THEN PLAY"N90
L19" ELSE PSET (X+6,160 H).15 A
$="N"+STR$(INT(H/2))+"L19"·PLAY
A$
4096 FOR I 1 TO 100·IF INKEY$ <
>CHR$(32) THEN NEXT I
4100 NEXT T
4130 LINE (250,195)-(5,180),4,B
F·PRINT#1,"NOVO ANGULO? (0 TERM
INA) " G$=""
4132 PRESET (220,180)·PRINT#1,G
$
4135 A$=INKEY$:IF A$=CHR$(13) T
HEN 4150
4140 IF A$<"0" OR A$>"9" THEN 4
135
4145 G$=G$+A$ GOTO 4132
4150 A VAL(G$)·IF A=0 THEN 4180
4160 IF A<0 OR A>89 THEN 4130
4170 GOTO 4060
4180 RETURN
```





```
4000  HGR : HCOLOR= 3
4020  HPLOT 0,0 TO 279,0 TO 279
,159 TO 0,159 TO 0,0
4040 SP = 50:A = 70
4060  HOME : VTAB 21: PRINT "AN
GULO =";A;" GRAUS  "
4065  VTAB 24: PRINT "TECLE <CR
 PARA DISPARAR ";
4070  GET A$: IF A$ <  >  CHR$
(13) THEN 4070
4075 AN = A *  ATN (1) / 45
4080  FOR T = 0 TO 250 STEP .5
4090 H = SP *  SIN (AN) * T - .
5 * 10 * T * T:X = SP *  COS (A
N) * T
4092  IF H < 0 THEN T = 250: GO
TO 4100
4093  IF X > 275 THEN T = 250
GOTO 4100
4094  IF H > 160 THEN  POKE 769
,200: POKE 768,2: CALL 770
4095  IF H < 160 THEN  HPLOT X
+ 2,160 - H: POKE 769,H: POKE 7
68,2: CALL 770
4097  POKE  - 16368,0
4098  FOR D = 1 TO 20: IF  PEEK
 (  - 16384) < 128 THEN  POKE
 - 16368,0: NEXT
4100  NEXT T
4130  HTAB 1: VTAB 23: CALL  -
958: VTAB 23: INPUT "NOVO ANGUL
O (0 TERMINA) ";A
4160  IF A < 0 OR A > 89 THEN 4
130
4170  IF A > 0 THEN 4060
4180  RETURN
```

A MAIOR DISTÂNCIA

Ao executar a quarta opção, você poderá ver a trajetória de um objeto lançado com uma velocidade inicial de 50 m/s e a uma inclinação de 70° (ambos os valores determinados na linha 4040 e usados na linha 4090).

A rotina funciona como as precedentes, exceto pelo fato de que se pode dar entrada a tantos ângulos quantos se desejar (entre 1° e 89°). Assim, toda vez que essa rotina for usada, a variável A da linha 4090 armazenará o ângulo que você escolheu. Desta vez, a rotina é um laço infinito, de modo que só se volta ao menu teclando-se 0 para um ângulo.

Usando esta rotina, tente encontrar o ângulo que lhe dá a maior distância de percurso nas direções vertical e horizontal. Será fácil descobri-lo: é o nosso conhecido ângulo de 45°.

Esse ângulo, porém, permite obter a maior distância apenas no caso em que os pontos de partida e de chegada estão no mesmo nível, pois é preciso levar em conta a resistência do ar.

ACASO E PROBABILIDADE

Um dos pioneiros da matemática moderna, o francês Blaise Pascal foi o criador de um triângulo de números extremamente importante para o cálculo das probabilidades.

A força de um computador está na sua capacidade de obedecer instruções repetidamente, com precisão e velocidade. Entretanto, quando comparado aos processos instantâneos de raciocínio do ser humano, o computador pode parecer lento e obsoleto. A mente humana é boa principalmente no que diz respeito a julgar e comparar parâmetros, tais como distância, velocidade e intensidade de luz.

Mas até o mais sofisticado dos órgãos comete erros ao calcular o resultado de eventos. Ainda assim, é essencial, por exemplo, poder dizer que, "para fins de seguro de vida, uma pessoa vive de 65 a 70 anos", ou que "não é provável que a terra volte a tremer no México em 1986". Tais afirmações são muito comuns no nosso dia-a-dia. Além disso, são importantes para fins sociais, comerciais e científicos. Assim, quando empregamos expressões como *prós e contras, estimativa, dúvida, expectativa*, estamos fazendo cálculos mentais de probabilidade.

A PROBABILIDADE EXPLICADA

Probabilidade é uma ferramenta científica para se medir o acaso. Usada para calcular o provável resultado de um evento, ela se baseia na existência de um número mensurável de resultados, como num jogo de futebol, ou num lançamento de dados, ou de moedas, num jogo de cartas etc. Naturalmente, temos que poder medir ou quantificar os resultados; por isso, eventos como corridas de cavalo ou jogos de futebol são assuntos difíceis para o cálculo de probabilidades. Quando dizemos "espero vencer", na verdade, estamos afirmando ser alta a *probabilidade* de conseguir a vitória e não apenas nosso desejo de triunfar.

A maioria das pessoas apóia-se na intuição como principal ferramenta no cálculo do acaso (ou probabilidade). Entretanto, se quisermos examinar os efeitos possíveis e, dentre eles, aqueles que são mais prováveis, num problema de probabilidade, podemos chegar a um resultado bastante preciso. Não é difícil aprender a prever os resultados mais prováveis, e mesmo que não possamos garanti-los, a possibilidade de acerto é bem maior que a de uma estimativa feita a esmo.

PROBABILIDADE E COMPUTAÇÃO

Qual será então a relação entre probabilidade e computação? Embora o tipo de probabilidade descrito até agora seja muito vago e dependente de estimativas, é possível deduzir fórmulas matemáticas para determinados acontecimentos que nos permitam prever os resultados mais prováveis com um certo grau de precisão.

Existem duas maneiras pelas quais o computador pode ser útil aqui. A primeira delas consiste em programá-lo para simular o próprio evento — é bem mais cômodo fazer com que a máquina jogue um dado 2000 vezes e depois analise o resultado do que termos nós mesmos que fazê-lo. A segunda maneira, entretanto, só funciona quando conhecemos as fórmulas para um certo evento; nesse caso, podemos fazer com que o computador calcule o resultado.

Visto por alguns como um exercício puramente teórico, esse tipo de cálculo serve de base para muitas aplicações úteis do computador. Uma dessas aplicações consiste nos jogos onde, por exemplo, os pontos ganhos dependem da probabilidade de certos resultados. Do mesmo modo, poderíamos escrever um programa que estabelecesse as probabilidades de chover num determinado dia. Por enquanto, porém, vamos concentrar nossa atenção apenas na teoria. Mais adiante, trataremos da construção de alguns desses aplicativos.



COMO MEDIR A PROBABILIDADE

A teoria de medição da probabilidade requer que sejam conhecidos os resultados possíveis de um evento, e que eles ocorram com uma certa frequência, passível de ser medida. A probabilidade de acontecer um certo evento é o número de vezes com que ele se repete (frequência), comparado com o total de todos os resultados possíveis. Ou, em outras palavras, é a frequência expressa em fração do conjunto dos resultados possíveis.

Devemos notar que, se um acontecimento se repetir toda vez que ocorrerem certas circunstâncias, então sua probabilidade será igual a 1. Isso porque sua frequência será igual ao número de resultados possíveis, e a divisão de dois números iguais resulta em 1. Temos, então, que 1 é a maior probabilidade possível, e que a soma das frações de probabilidade de todos os resultados possíveis resultará invariavelmente nesse número.

Um dos métodos mais simples e antigos de visualizar a probabilidade é lançar uma moeda e prever o resultado, ou seja, que face terá ao cair: cara ou coroa. Como a moeda tem só dois lados, podemos inferir que, qualquer que fosse o número de lançamentos, deveríamos obter cara, metade das vezes, e coroa a outra metade, ignorando a remota possibilidade de a moeda cair em pé. Para ilustrar melhor esse método, digite e rode o primeiro programa:



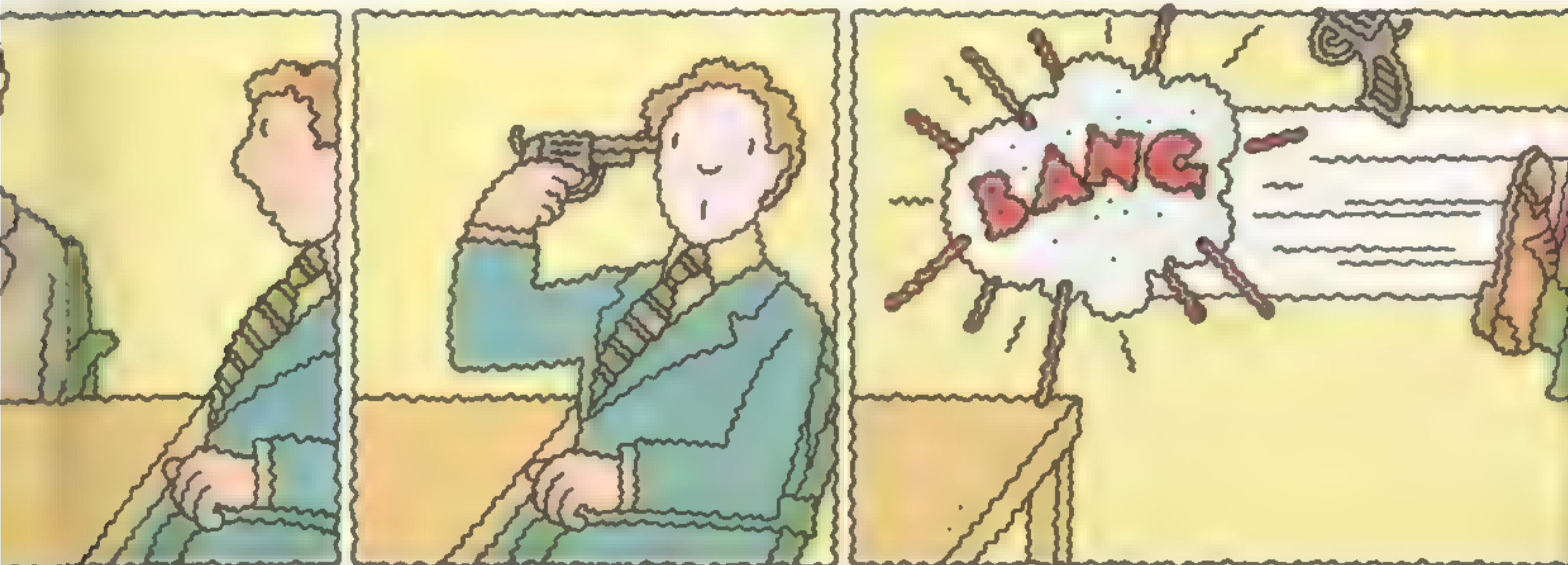
```
10 COLOR 1,3,3:CLS
20 INPUT"QUAL O TESTE (1-4) ";X
30 CLS IF X<1 OR X>4 THEN 10
40 ON X GOTO 70,70,180,460
50 REM . ...  ..probabilidade
60 REM...lançamento de moedas
70 H=0:T=0
80 LOCATE5,3:PRINT"PRESSIONE A TECLA DE ESPAÇO"
85 LOCATE9,4 PRINT"PARA LANÇAR A MOEDA"
90 LOCATE13,7:PRINT"CARAS:  0":PRINTTAB(13)"COROAS: 0"
100 A$=INKEY$:IFA$=CHR$(13) THEN SCREEN0 RUN
105 IFA$<>" "THEN 100
110 IF X=2 THEN FORN=1 TO 100
120 IFINT(RND(1)*2)+1=1 THEN H=H+1 ELSE T=T+1
130 LOCATE20,7:PRINTH:LOCATE20,8 PRINTT
140 IF X=1 THEN 100
160 NEXTN:GOTO 100
```



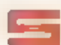
```
5   HOME
20  INPUT "QUAL O TESTE (1-4) ? ";X
30  HOME   IF X < 1 OR X > 4 THEN 5
40  ON X GOTO 70,70,180,460
50  REM  .  ... PROBABILIDADE
60  REM ..LANCAMENTO DE MOEDAS
70 H = 0:T = 0
80  PRINT "PRESSIONE A TECLA DE ESPACO"
85  PRINT "    PARA LANCAR A MOEDA"
90  VTAB (4): PRINT "CARAS:  0": PRINT "COROAS: 0"
100  GET A$: IF A$ =  CHR$ (13) THEN  TEXT : RUN
105  IF A$ <  > " " THEN 100
110  IF X = 2 THEN  FOR N = 1 TO 100
120  IF  INT ( RND (1) * 2) + 1 = 1 THEN H = H + 1: GOTO 130
125 T = T + 1
130  VTAB (4): HTAB (9): PRINT ;H: HTAB (9): PRINT ;T
140  IF X = 1 THEN 100
160  NEXT N: GOTO 100
```



```
10 PMODE 3,1:CLS
20 INPUT"QUAL O TESTE (1-4) ";X
30 CLS:IF X<1 OR X>4 THEN 20
40 ON X GOTO 70,70,180,460
50 REM .... PROBABILIDADE
60 REM .... LANCAMENTO DA MOEDA
70 H=0:T=0
```



```
80 PRINT @65,"PRESSIONE A BARRA
 DE ESPACOS     PARA LANCAR A MO
EDA"
90 PRINT @288,"CARAS - 0":PRINT
"COROAS- 0"
100 A$=INKEY$:IF A$<>" " THEN 1
00
110 IF X=2 THEN FOR N=1 TO 100
120 IF RND(2)=1 THEN H=H+1:PRIN
T @204,"CA":PRINT @295,H:ELSE T
=T+1:PRINT @207,"CO":PRINT @327
,T:
130 IF X=1 AND PEEK(345)<>247 T
HEN 130
140 PRINT @204,""
150 IF X=1 THEN FOR D=0 TO 400:
NEXT:GOTO 120 ELSE NEXT
160 A$=INKEY$:IF A$<>CHR$(13) T
HEN 160 ELSE END
```

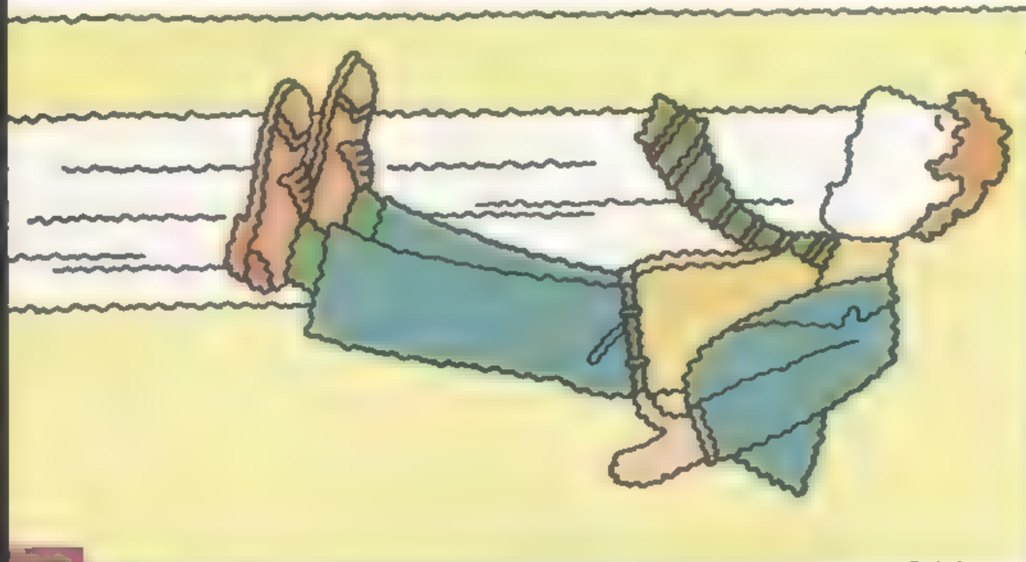


```
  5 BORDER 7  PAPER 7: INK 9:
  CLS
 10 DIM n(4)
 20 RESTORE 9000: FOR n=1 TO 4
 : READ n(n): NEXT n
 30 INPUT "Qual o teste (1 a 4
 )?",x  CLS
 40 BORDER x: GOTO n(x)
 50 REM Probabilidade
 60 REM Lancamento de Moeda
 70 LET h=0: LET t=0
 80 PRINT AT 2,1;"Pressione <S
 PACE> para lancar"
 90 PRINT AT 20,10;"CARAS - 0"
 ,AT 21,10;"COROAS  0"
100 IF INKEY$<>CHR$ 32 THEN
GOTO 100
110 IF x=2 THEN  FOR n=1 TO
 100
120 IF INT (RND*2)=1 THEN  LET
h=h+1: PRINT AT 10,15,"CA";AT
20,18,h: GOTO 130
125 LET t t+1  PRINT AT 12,15;
"CO";AT 21,18;t
130 IF x=1 THEN  IF INKEY$<>
CHR$ 32 THEN  GOTO 130
140 PRINT AT 10,15;"  ";AT 12,
15;"  "
150 IF x=1 THEN  FOR m=1 TO
 100: NEXT m: GOTO 120
155 NEXT n: STOP
9000 DATA 70,70,170,460
```

Este programa será incrementado mais adiante. Ao rodá-lo, seremos perguntados a respeito do número do teste que desejamos fazer. Esse primeiro programa contém somente os dois testes iniciais. Digite 1 e estaremos prontos para lançar a moeda — usando a tecla de espaço. O ponto crucial do programa está na linha 120, que gera aleatoriamente valores 1 (para cara) e 0 (para coroa), além de fazer a contagem do número de caras e coroas obtidas). A linha 150 no programa do Spectrum e também no do TRS-Color provoca uma pausa entre as jogadas.

Poucos lançamentos, provavelmente, fornecerão valores diferentes para cara e coroa, representados no programa pelas variáveis **H** e **T**, respectivamente. Quanto maior o número de jogadas, mais próximas uma da outra serão as freqüências de **H** e **T** — tendendo, cada uma, à metade do número total de lançamentos (ou 50%). Para demonstrar esse efeito, rode o programa novamente, mas, desta vez, selecione o teste 2. Ao ser pressionada a tecla de espaço, um laço preparado pela linha 110 lançará a moeda cem vezes. Note que as duas variáveis, **H** e **T**, estão bem próximas de 50. Mude o 100 da linha 110 para 1000, rode o programa e observe como cara e coroa tendem para 500.

Pode acontecer que, num teste de poucas jogadas, todas elas sejam coroa. Apesar disso, a probabilidade de se obter coroa será sempre igual a 1/2 (meio). Convém não nos esquecermos disso, principalmente quando estivermos trabalhando com mais de um evento. Muitos acreditam que, se dez lançamentos seguidos de uma moeda derem cara, a possibilidade de se obter coroa na décima-primeira jogada será maior do que antes. Isso não é verdade; a possibilidade de dar coroa ainda será de 50%. Os eventos passados não têm influência sobre os lançamentos seguintes. No entanto, se lançarmos onze moedas ao mesmo tempo, a chance de se obter onze coroas será menor que a de obter dez coroas e uma cara. A maior chance é de que o número de coroas seja próximo ao número de caras.



MÚLTIPLOS EVENTOS

Quando há vários eventos, algumas informações adicionais são necessárias para se poder prever a probabilidade de cada resultado. Uma informação essencial é o número total de resultados possíveis. Por exemplo, se lançarmos duas vezes uma moeda, teremos três resultados possíveis: duas caras, uma cara e uma coroa, e duas coroas. À primeira vista, pareceria que cada resultado pode ocorrer 1/3 das vezes. Na verdade, as probabilidades são duas caras acontecerem 1/4 (25%) das vezes; duas coroas, 1/4; e uma cara e uma coroa, 1/2. Para entender essa terceira probabilidade, precisamos de outra informação essencial, que dê conta de quantas vezes cada resultado ocorre. Uma cara e uma coroa acontecem duas vezes, porque existem duas maneiras de se obter tal resultado: uma cara e depois uma coroa, ou uma coroa e depois uma cara. Conseguimos assim um total de quatro resultados, três dos quais diferentes entre si.

Na prática, costuma-se usar dois recursos matemáticos que nos poupam o trabalho de calcular a ocorrência de cada resultado: o teorema binomial e o triângulo de Pascal. Binomial significa "dois termos". Se um evento tem só dois resultados possíveis e conhecemos a probabilidade de cada um deles, podemos usar o teorema binomial para calcular tais probabilidades. O teorema binomial nos diz o que devemos esperar de testes repetidos de um evento com dois resultados possíveis. Chamaremos **P** à probabilidade de um dos resultados e **Q** à do outro (lembremos que a soma de **P** e **Q** deve ser igual a 1). Denominaremos **N** o número de eventos.

No exemplo do lançamento de uma moeda, **P** (digamos, a possibilidade de dar cara) e **Q** (coroa) serão iguais a 1/2 para cada jogada. De acordo com o teo-

rema binomial, a chance de que qualquer evento aconteça duas vezes seguidas é a probabilidade de que ocorra uma vez multiplicada por ela mesma. Em geral, a regra consiste na probabilidade do evento elevado à potência **N**. Portanto, para o caso de duas caras seguidas, **P↑N = 1/2 x 1/2 = 1/4**. Existe então uma chance em quatro de que obtenhamos duas caras seguidas. Do mesmo modo, a possibilidade de se conseguir cinco coroas sucessivas é **P↑N**=(1/2 ↑ 5), ou 1/32.

Como veremos mais tarde, podemos usar esse método para calcular a probabilidade em qualquer caso onde existam somente dois resultados possíveis (sim/não ou cara/coroa, por exemplo). Mas qual seria a chance de se obterem três caras e duas coroas em cinco lançamentos? Para responder a essa pergunta, precisamos de um modelo mais complexo. Muito útil neste caso é o triângulo de números, desenvolvido pelo matemático, físico e filósofo francês Blaise Pascal. Esse triângulo fornece todos os resultados possíveis de um evento binomial, e pode ser entendido como algumas fileiras de numeros. As primeiras sete fileiras são:

```
Fileira 0)                 1
Fileira 1)              1     1
Fileira 2)           1     2     1
Fileira 3)        1     3     3     1
Fileira 4)      1    4     6     4    1
Fileira 5)    1   5    10    10    5   1
Fileira 6) 1   6   15    20    15   6   1
```



Para construir um triângulo como esse, escreva primeiramente as fileiras 0 e 1, as mais fáceis de serem lembradas. A fileira 2 começa com um 1 à esquerda da fileira 1 e termina com um 1 à direita. O número do meio é obtido pela soma dos dois algarismos imediatamente acima dele (1+1). Da mesma maneira, a fileira 6 é obtida somando-se 5+1, 5+10, 10+10, 10+5 e 5+1. Se continuarmos com esse processo, teremos um triângulo com um número de fileiras cada vez maior, o que seria difícil de conseguir com a utilização de qualquer outro metodo

O triângulo de Pascal nos dá todas as informações de que precisamos quando lançamos diversas moedas (ou uma moeda várias vezes). O número de moedas (ou de lançamentos) estabelece o número da fileira que devemos olhar; o número de itens na fileira determina o de resultados diferentes. Por exemplo, existem duas possibilidades para uma moeda (1 e 1 na fileira 1) e sete para seis moedas (1, 6, 15, 20, 15, 6, 1 na fileira 6). A soma dos itens na fileira nos dá o número total de resultados (2 para uma moeda, 4 para duas moedas, e assim por diante).

Cada número na fileira define a probabilidade. Por exemplo, na fileira 2, o primeiro número (1) é a probabilidade para duas caras; o segundo (2), a probabilidade para uma cara e uma coroa; o terceiro (1), para duas coroas. Evidentemente, a freqüência tem que ser dividida pelo número total de resultados (4, neste caso) para se obter a probabilidade. Note que o resultado da adição dos números em cada fileira é sempre uma potência de dois (1, 2, 4, 8, 16). Isso acontece porque, para qualquer um dos eventos, há só duas opções: cara ou coroa.

A praticidade desse método é posta à prova quando queremos, por exemplo, calcular as probabilidades de lançamento de trinta moedas, a construção de um triângulo com trinta fileiras seria muito trabalhosa, além de tomar muito espaço. Existe, porém, um método gráfico ao qual podemos recorrer em tais situações, e é aqui que entra a ajuda do computador.

CURVAS DE DISTRIBUIÇÃO

Onde existirem muitos resultados com probabilidades não muito claras, conseguiremos quase sempre calcular o que precisamos, plotando (ou seja, marcando num gráfico) uma curva de distribuição, definida pela frequência dos resultados que conhecemos. Como em qualquer método gráfico, as informações nele contidas são percebidas logo de início. Se, por exemplo, uma moeda for lançada trinta vezes (o que seria o mesmo que lançar trinta moedas de uma só vez), podemos traçar um gráfico com o número de caras obtidas. Digite a pró-

xima seção de programa, mas não apague a anterior:



```
170 REM  . ..picos randômicos
180 SCREEN2:FORX=5 TO 255 STEP
[illegible]
190 GM=0·GOSUB 610
200 FORY=0 TO H*6 STEP 2
210 PSET(X,188-Y),1
220 NEXT Y,X
230 IFINKEY$=""THEN230 ELSE RUN
610 H=0:T=0
650 FORS=1 TO 30
660 IFINT(RND(1)*2)+1=1 THEN H=
H+1 ELSE T=T+1
670 NEXT
680 RETURN
```



```
170  REM .... PICOS ALEATORIOS
180  HGR : HCOLOR= 3: FOR X = 4
 TO 275 STEP 10
190 GM = 0: GOSUB 610
200  FOR N = 0 TO H· HPLOT X,16
0 - (N * 6): NEXT N
220  NEXT X
230  GET A$: IF A$ = "" THEN 23
0
235  TEXT   RUN
610  REM ...   LANCAMENTO
620 H = 0:T = 0
630  HOME · VTAB (22): PRINT "C
ARAS:  ": PRINT "COROAS: "
640  IF GM <  > 0 THEN  VTAB (2
2)· HTAB (31)· PRINT "JOGO· ",G
M· HTAB (17)· PRINT "CARAS EM 3
0 LANCAMENTOS"
650  FOR S = 1 TO 30
660  IF  INT ( RND (1) * 2) + 1
 = 1 THEN H = H + 1· GOTO 668
665 T = T + 1
668  VTAB (22)· HTAB (10)· PRIN
T ;H: HTAB (10): PRINT ;T
670  NEXT S
680  RETURN
```



```
170 REM .... GRAFICO
180 PCLS4:SCREEN 1,0:FOR X=0 TO
 255 STEP 4
190 GM=0:GOSUB 610
200 FOR Y=0 TO H*6 STEP 2
210 PSET(X,188-Y,2)
220 NEXT Y,X
230 GOTO 160
610 H=0:T=0
650 FOR TS=1 TO 30
660 IF RND(2)-1 THEN H=H+1 ELSE
 T=T+1
670 NEXT
680 RETURN
```



```
170 REM Grafico
175 PLOT 0,0: DRAW 180,0
180 FOR x=4 TO 160 STEP 4
190 LET gm=0: GOSUB 610
200 FOR n=0 TO h: PLOT x,n*6:
NEXT n
220 NEXT x
230 STOP
610 REM Lancamento
620 LET h=0: LET t=0
630 PRINT AT 4,22;"CARAS -";h;
AT 6,22;"COROAS-";t
640 IF gm<>0 THEN  PRINT AT 0,
0;"JOGADAS-";gm;AT 21,3;"CARAS
EM 30 LANCAMENTOS·"
650 FOR s=1 TO 30
660 IF RND>=.5 THEN  LET h=h+1
: PRINT AT 2,24;"  CA";AT 4,29
;h;" ": GOTO 670
665 LET t=t+1  PRINT AT 2,24,"
CO  ";AT 6,29;t;" "
670 NEXT s
680 RETURN
```

Ao rodar o programa, desta vez, selecione o teste 3. Devemos ver na tela um gráfico com uma série de pontos chegando a vários picos. Esta é uma das muitas formas possíveis nesse tipo de análise. Os picos são os números de caras em cada trinta lançamentos ao longo do eixo Y, espaçados igualmente ao longo do eixo X. Existem mais picos altos do que baixos. A razão disso é que a possibilidade de se obter cerca de quinze (ou de doze a dezessete) caras é muito maior que a de se obter um número menor ou maior. Isso também pode ser observado no triângulo de Pascal, onde os valores mais altos se encontram na região central.

A linha 180 gera um laço para espaçar os pontos ao longo do eixo X. A variável **GM** (número de jogadas) é zerada na linha 190 e uma rotina (linhas 610 a 680) é chamada para executar cada bateria de trinta lançamentos. Essa rotina usa os elementos do segundo teste, mas lança a moeda "eletrônica" trinta vezes e não cem. Com exceção do TRS-COLOR e do MSX, ao rodarmos esse teste, notaremos o placar para cara e coroa que aparecerá na tela. Uma vez atingidos os trinta lançamentos, o número de caras que foi acumulado na rotina sofrerá um ajuste de escala na linha 200 e será plotado na coordenada Y pela linha 210 (linha 200, no Spectrum e no Apple).

Para tirar o máximo proveito de uma análise desse tipo, precisamos rearranjar as informações, de modo que obtenhamos uma das curvas mais conhecidas no meio estatístico: a distribuição normal. Digite mais estas linhas para obtermos tal curva:



```
450 REM.. distribuição normal
460 DIM G(30)
470 CLS SCREEN2 LINE(6,0) (6.19
1).15·LINE (255.191).15
480 GOSUB 560
485 AS=INKEYS·IFAS<>" "THEN 485
560 DEF FNN(X)=1/(4 4429*2 718
^(X*X)/2))
570 DRAW"BM7,190":FORX=2 TO 255
 STEP2
580 LINE-(X,191-640*FNN((X 127)
/24)).1
590 NEXT RETURN
```



```
450  REM   DISTRIB NORMAL
460  DIM G(30)· HGR   HCOLOR= 3

470  HPLOT 0,0 TO 0,159  HPLOT
 TO 279.159
480  GOSUB 560
485  GET AS  IF AS <  > " " THE
N 485
490  END
560  REM      GRAFICO
565  DEF  FN N(X) = 1 / (4.4429
 * 2 718   ((X * X) / 2))
570  FOR X = 1 TO 280 STEP 2
580  HPLOT X,158   660 *  FN N(
(X - 140) / 24)
590  NEXT X: RETURN
```



```
450 REM .... DISTR.NORMAL
460 DIM G(30)
470 PCLS:SCREEN 1,0:LINE(0,0)-(
0,191),PSET:LINE-(255,191),PSET
480 GOSUB 560
485 AS=INKEYS:IF AS<>" " THEN 4
85
560 DEFFND(X)=1/(4.4429*2.718^(
(X*X)/2))
570 COLOR 2,3·DRAW"BM2,190":FOR
 X=2 TO 255 STEP 2
580 LINE-(X,191-640*FND((X-127)
/24)),PSET
590 NEXT:RETURN
```



```
450 REM Distr Normal
460 DIM g(30)
470 PLOT 4,150: DRAW 0,-140:
DRAW 245,0
480 GOSUB 560
485 IF INKEYS<>CHRS 32 THEN
GOTO 485
560 REM Graf
570 PLOT 4,10  FOR x=0 TO 1200
STEP 20
580 DRAW 4,600*FN n(ABS ((x-
600,/140))+10-PEEK 23678
590 NEXT x· RETURN
600 DEF FN n(x)=1/(PI*1 4142*
2 718^((x^2)/2))
```

Desta vez, rode o programa e selecione o teste 4, que é o de uma curva de distribuição do tipo normal. A linha 460 dimensiona uma matriz que será usada posteriormente na contagem das caras obtidas.

A linha 470 desenha os eixos das coordenadas X e Y e a linha 480 chama a rotina que desenha a curva. Essa rotina usa uma função matemática (linha 580) para desenhar a curva, o que explica a forma "perfeita" desta última. A função é definida pelas linhas 560 (MSX e TRS-Color), 565 (Apple e TK-2000) e 600 (Spectrum). Por enquanto, porém, não aperte nenhuma tecla, pois a rotina está incompleta. Aguarde o desenvolvimento do programa.

Quando usamos informações reais, torna-se difícil plotar uma curva contínua. Isso já era de se esperar, pois estamos lidando com probabilidades e não com certezas. A probabilidade de um resultado, tal como o de chover na Índia na época das monções, pode ser alta; mas, no caso do exemplo, têm sido registrados períodos em que a seca toma o lugar das chuvas.

O próximo teste ilustra muito bem esse ponto. Ele repete várias vezes os experimentos anteriores das moedas e coloca no gráfico (plota) os resultados obtidos. Digite então a segunda parte do teste 4:



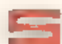
```
490 FOR GM=1 TO 500
500 GOSUB 610
510 G(H)=G(H)+1
520 PSET(H*8+7,192-G(H)*2),1
530 NEXT
540 GOTO 230
```



```
490  FOR GM = 1 TO 200
500  GOSUB 610
510 G(H) = G(H) + 1
520  HPLOT 20 + 8 * H,160   G(H
) * 4
530  NEXT GM
540  GOTO 230
```



```
490 FOR GM=1 TO 500
500 GOSUB 610
510 G(H)=G(H)+1
520 PSET(H*8+7,192-G(H)*2.3)
530 NEXT
540 GOTO 160
```



```
490 FOR g=1 TO 200  LET gm=g
500 GOSUB 610
510 LET g(h)=g(h)+1
520 PLOT 8+8*h,10+4*g(h)
530 PRINT AT 21,27,h," "
540 NEXT g
550 STOP
```

Agora, rode novamente o quarto teste. Quando a curva ideal tiver sido desenhada, pressione a tecla de espaço para iniciar o lançamento. Aparecerá então uma série de pontos "crescendo" para preencher o espaço compreendido pela curva. Quando o teste estiver completado, quinhentos pontos (duzentos no Spectrum e no Apple) terão sido colocados no gráfico (especificados na linha 490). Em alguns micros, tal como no Spectrum, o tempo de execução desse teste é de alguns minutos. Para acelerá-lo, costuma-se inserir desvios (**GOTO**) em lugares apropriados ao longo do programa, mas na maioria das vezes isso traz alteração no funcionamento dos outros testes.

Esta seção do programa chama a rotina (linha 500) que lança a moeda trinta vezes (no Apple e no Spectrum, os lançamentos são mostrados na tela, como no terceiro teste). Cada bateria de trinta lançamentos forma um jogo, e pode resultar em certo número de caras entre 0 e 30. Portanto, na matriz da linha 460, **H** pode variar de 0 a 30. Se analisarmos pela óptica da lei das probabilidades, veremos que muito dificilmente **H** assumiria valor 0 ou valor 30 numa bateria de trinta lançamentos. Uma situação assim é conhecida como "ocorrência rara", justamente pela maneira com que os números aleatórios são gerados. Na verdade, números extremamente pequenos ou extremamente grandes de caras em lançamentos como esse não acontecem com frequência. Sua ocorrência é *possível*, mas a probabilidade é muito baixa.

A linha 510 faz a contagem do resultado de cada jogo por intermédio de uma matriz. Por exemplo, toda vez que o resultado de um determinado jogo for onze caras, o elemento de matriz **G(11)** será incrementado de 1. Da mesma maneira, toda vez que o resultado for quinze caras, **G(15)** será também incrementado de 1. Inicialmente, todos os elementos da matriz são iguais a zero.

Após cada jogada, a linha 520 altera a escala do valor de **H** (número de caras em trinta lançamentos), a fim de compatibilizá-lo com os eixos X e Y. A próxima vez que ocorrer o mesmo resultado, um ponto será plotado na mesma coordenada X, mas uma unidade acima na coordenada Y.

A linha 530 faz a contagem do **H** em cada jogada, verificando também o número de jogadas.

COMO USAR A CURVA

Rode algumas vezes o teste 4 para ter uma idéia de como variam os pontos sob a curva; depois, rode-o com menores valores finais para **GM** na linha 490. Mesmo sem a curva ideal, seremos logo capazes de imaginar uma curva idealizada passando pelos picos. Na prática, o inverso desse processo imaginário é extremamente importante, pois quando conhecemos o perfil de uma curva podemos fazer a previsão do resultado de testes futuros.

O valor de **H** no pico central tem um interesse especial. Ele é a média dos 31 possíveis valores ao longo do eixo X (neste caso, 15). A média identifica o pico da curva, e constitui também o valor mais provável; sozinha, porém, ela não é muito informativa. Embora possamos dizer que 15 seja o resultado mais provável, 14 e 16 são quase tão prováveis quanto ele. Existe uma faixa de valores comuns em torno do pico, e é importante conhecer a largura desta faixa. A média é usada na especificação de um outro parâmetro estatístico importante — o desvio padrão —, que é a medida da largura da faixa de pontos comuns. A fórmula para o desvio padrão é relativamente complicada. Uma vez calculado esse parâmetro, podemos atribuir uma probabilidade para qualquer um dos pontos na curva.

O desvio padrão mede o quanto os valores variam nos dois lados da média. Por exemplo, uma seção de curva com desvio padrão de 1,96 em cada lado da média englobará 95% dos resultados. Se aumentarmos o desvio padrão para 2,58, a curva passará a conter 99% dos resultados.

Os pacotes comerciais de software estatístico calculam o desvio padrão com grande facilidade e rapidez.

SEIS RESULTADOS POSSÍVEIS

Existem muitos casos de eventos onde temos mais do que dois resultados possíveis. Em tais casos, calcular a chance de algum evento não é tão simples como olhar uma fileira do triângulo de Pascal.

Por exemplo, quando jogamos um dado, podemos conseguir um entre seis resultados. Se o dado não for viciado, todos os resultados são equiprováveis, ou seja, têm a mesma possibilidade de vir a acontecer.

Os resultados para o lançamento de dois dados podem ser calculados por intermédio da construção de uma tabela; mas, quanto mais resultados tivermos, mais complicado ficará o método que teremos de seguir.

Aqui está uma tabela com todos os resultados possíveis no lançamento de um par de dados:

		Valor do primeiro dado					
		1	2	3	4	5	6
Valor do segundo dado	1	2	3	4	5	6	7
	2	3	4	5	6	7	8
	3	4	5	6	7	8	9
	4	5	6	7	8	9	10
	5	6	7	8	9	10	11
	6	7	8	9	10	11	12

Como vemos na tabela, existem 36 maneiras diferentes de os dados caírem (seis linhas vezes seis colunas), embora haja somente onze resultados diferentes. A tabela apresenta várias outras informações. Existe uma possibilidade em 36 de obtermos os pontos mínimo e máximo (2 ou 12 aparecem somente uma vez na tabela); em contrapartida, há seis possibilidades em 36 (1/6) de obtermos 7 como resultado. Existe também uma chance em 36 de jogarmos repetido. Essa chance é mostrada na diagonal que vai de 2 (duas vezes o número 1) até 12 (dois 6).

Combinando o teorema binomial com essa tabela, podemos calcular as probabilidades para muitos jogos. Um bom exemplo de evento de múltiplo lançamento de dados é o jogo Monopólio, onde o jogador, caso esteja na "cadeia", tem direito a três lances para tentar conseguir uma dupla (dois dados iguais). Intuitivamente, pode parecer que ele conta com 50/50 de possibilidades de sair da "cadeia" (três jogadas, cada uma com 1/6 de chance), mas isso não é verdade. Pela tabela, podemos verificar que a chance de o jogador *não* conseguir uma dupla é de 30/36 (5/6) para cada jogada. Usando o teorema binomial, perceberemos facilmente que a chance de ele não conseguir uma dupla três vezes seguidas é 5/6 elevado à potência 3, ou 125/216. Isso equivale, aproximadamente, a 58% de possibilidades de fracasso.

SIMULAÇÕES ESPACIAIS

- ESCOLHA DA TRAJETÓRIA
- AVALIAÇÃO DA DISTÂNCIA
- CIRCUNDANDO A TERRA
- MANOBRAS NO ESPAÇO
- MOVIMENTO PLANETÁRIO

Atingir as estrelas é um antigo sonho do homem. Com nossos programas, que permitem simular a trajetória de um corpo no espaço, você estará mais perto da realização desse sonho.

No artigo sobre trajetórias (página 766), vimos como a velocidade de um projétil pode ser separada, para fins de análise, em uma componente vertical e outra horizontal. Vimos também como a distância do ponto de partida ao ponto de chegada pode mudar conforme o ângulo de lançamento e a velocidade inicial. Neste artigo, avançamos mais um pouco no estudo dos corpos em movimento. Analisamos objetos que se movem em campos de baixa gravidade, simulando sua trajetória a partir de pequenas distâncias da superfície da Terra e estudando corpos em órbita.

Digite este primeiro programa. Ele é um bom exemplo de como utilizar seu conhecimento de trajetórias para montar jogos que usam disparos ou outros lançamentos mais desafiadores.



```
10 FOR n=0 TO 31· READ a·
POKE USR "a"+n,a: NEXT n
```



```
20 BORDER 4: PAPER 0: INK 9:
CLS
70 LET a=INT (RND*5): LET b=
INT (RND*5)+26: LET c=INT (
RND*8)+2· LET h=INT (RND*8)+2
: LET at=INT (RND*100-c*8)+1·
 LET d=0
90 LET d=d+1
100 CLS . GOSUB 300
110 INPUT "ANGULO ?",a2
120 IF a2>89 OR a2<1 THEN
GOTO 110
130 INPUT "VELOCIDADE ?",e
140 IF e=0 THEN  GOTO 100
160 LET an=a2*(PI/180)· LET
x=0
170 LET x2=x+(a+1)*8
180 LET y=8+(x*TAN an-4*x*x/(e
```



```
*e*COS an*COS an))
185 IF ATTR (21-INT (y/8),INT
(x2/8))=6 THEN  GOTO 245
190 IF y<=0 THEN  GOTO 245
200 IF (y>175 OR x2>255) AND d
<10 THEN  GOTO 90
205 IF y>175 OR x2>255 THEN
GOTO 270
210 PLOT  INK 8;x2,y: SOUND
.01,y/10
220 LET x=x+3
230 GOTO 170
245 IF x2>=b*8+1 AND x2<=b*8+
10 THEN  PRINT AT 21,b;CHR$
145: FOR n=20 TO 0 STEP -1:
SOUND .01,n· NEXT n: GOTO 270
246 IF d<10 THEN  GOTO 90
270 IF d=10 THEN  PRINT AT 8,
```



```
10;"ERROU !": GOTO 280
275 PRINT  INVERSE 1;AT 8,10;"
BOM TIRO !";AT 10,5;"VOCE CONS
EGUIU APOS ";d
280 PAUSE 100· PRINT  BRIGHT 1
: PAPER 2: INK 6;AT 13,8;"OUTR
A VEZ?": PAUSE 200
290 GOTO 70
300 PRINT  INK 5;AT 21,a;CHR$
144;AT 21,b;CHR$ 146
310 FOR n=1 TO c: PRINT AT 21-
n+1,12,· FOR m=1 TO c· PRINT
INK 6;CHR$ 147;· NEXT m· NEXT
 n
320 RETURN
500 DATA 3,6,60,40,104,60,126,
255
510 DATA 36,90,165,90,60,155,
24,60
520 DATA 24,36,66,153,153,66,
36,127
530 DATA 28,42,85,170,127,170,
85,255
```



```
10 PMODE 4:DIM G(1),E(1),T(1),B
(1)
20 FOR K=1536 TO 2528 STEP 32:R
EAD A:POKE K,A:NEXT
30 GET (0,0)-(7,7),G,G:GET(0,8)
-(7,15),E,G
40 GET (0,16)-(7,23),T,G:GET(0,
24)-(7,31),B,G
50 DATA 3,6,60,40,104,60,126,25
5,36,90,165,90,60,155,24,60
60 DATA 24,36,66,153,153,66,36,
```

```
127,28,42,85,170,127,170,85,255
70 A=RND(51)-1:B=RND(51)+197:C=
RND(8):H=RND(8)+1:ST=RND(100-C*
8)+70:D=0:CLS
80 D=D+1
90 PCLS:SCREEN 1,1:GOSUB 320
100 IF INKEY$="" THEN 100
110 PRINT:INPUT " ANGULO ";A2
120 IF A2>89 OR A2<1 THEN 110
130 INPUT " VELOCIDADE ";E
140 IF E=0 THEN 130
150 SCREEN 1,1
160 AN=A2*ATN(1)/45:X=0
170 X2=X+A+8
180 Y=183-(X*TAN(AN)-4*X*X/(E*E
*COS(AN)*COS(AN)))
190 IF Y>190 THEN 250
200 IF X2>=ST AND X2<ST+C*8+7 A
ND Y>183-H*8 THEN 250
210 IF Y>0 THEN PSET(X2,Y,5):SO
UND 200-Y,1 ELSE SOUND 255 AND
(200-Y),1
220 X=X+3
230 IF X2<255 THEN 170
240 GOTO 290
250 IF Y>190 THEN Y=184
260 PUT(X2-4,Y)-(X2+3,Y+7),E,PS
ET:PLAY"T5001ADECBFGAEDBGDE"
270 PUT(A,184)-(A+7,191),G,PSET
:PUT(B,184)-(B+7,191),T,PSET
280 IF X2>=B AND X2<B+7 THEN 30
0
290 IF D<10 THEN 80
300 CLS:IF D=10 THEN PRINT @41,
" ERROU !" ELSE PRINT @41,"BOM
TIRO !":PRINT @164," VOCE CONSE
GUIU APOS";D;"DISPAROS."
310 FOR W=1 TO 3000:NEXT:PRINT:
PRINT" OUTRA VEZ ...":FOR W=1 T
O 1200:NEXT:GOTO 70
320 FOR X=0 TO C:FOR Y=0 TO H:P
UT(ST+X*8,184-8*Y)-(ST+X*8+7,19
1-8*Y),B,PSET
330 NEXT Y,X
340 PUT(A,184)-(A+7,191),G,PSET
:PUT(B,184)-(B+7,191),T,PSET:RE
TURN
```



```
10 W=RND(-TIME)
70 A=INT(RND(1)*51):B=INT(RND(1
)*51+198):C=INT(RND(1)*8+1):H=I
NT(RND(1)*9)+2:ST=INT(RND(1)*(1
01-C*8))+70:D=0:CLS
80 D=D+1
90 SCREEN2·GOSUB 320
100 IF INKEY$="" THEN 100
110 SCREEN0:PRINT:INPUT"Angulo
";A2
120 IF A2>89 OR A2<1 THEN 110
130 PRINT:INPUT"Velocidade ";E
140 IF E=0 THEN 130
150 SCREEN2:GOSUB 320
160 AN=A2*ATN(1)/45:X=0
170 X2=X+A+8
180 Y=183-(X*TAN(AN)-4*X*X/(E*E
*COS(AN)*COS(AN)))
190 IF Y>190 THEN 250
200 IF X2>=ST AND X2<ST+C*8+7 A
ND Y>183-H*8 THEN 250
```



```
210 IF Y>0 THEN PSET(X2,Y):A$="
N"+STR$(INT((190-Y)/2))+"L19":P
LAY A$
220 X=X+3
230 IF X2<255 THEN 170
240 GOTO 290
250 IF Y>190 THEN Y=184
260 LINE (X2,Y)-(X2,Y-8):PLAY "
O1AAA"
280 IF X2>=B AND X2<B+7 THEN 30
0
290 FOR W=1 TO 1500:NEXT:IF D<1
0 THEN 80
300 SCREEN0:IF D=10 THEN PRINT:
PRINT"TUDO FOI EM VÃO!" ELSE PR
INT:PRINT"BOM TIRO!":PRINT"Você
 acertou em";D;"disparos!"
310 FOR W=1 TO 3000:NEXT:PRINT:
PRINT"Outra vez...":FOR W=1 TO
1500:NEXT·GOTO 70
320 FOR X=0 TO C:FOR Y=0 TO H:P
SET (ST+X*8,184-Y*8)
330 NEXT Y,X
340 LINE (A,184)-(A+5,184):LINE
 (A+3,184)-(A+3,180):LINE (B,18
4)-(B+5,184):LINE (B+3,184)-(B+
3,180):RETURN
```



```
10  FOR I = 770 TO 795: READ A:
 POKE I,A: NEXT
20  POKE 768,2
70 A =  INT ( RND (1) * 51):B =
  INT ( RND (1) * 51 + 198):C =
  INT ( RND (1) * 8) + 1:H =  I
NT ( RND (1) * 9) + 2:ST =  INT
 ( RND (1) * (101 - C * 8)) + 7
0:D = 0: HOME : VTAB 21
80 D = D + 1
90  HGR : HCOLOR= 3: GOSUB 320
100  GET A$
110  INPUT "ANGULO? ";A2
120  IF A2 > 89 OR A2 < 1 THEN
110
130  VTAB  PEEK (37): HTAB 20:
INPUT "VELOCIDADE? ";E
140  IF E = 0 THEN 130
160 AN = A2 *  ATN (1) / 45:X =
 0
170 X2 = X + A + 8
180 Y = 157 - (X *  TAN (AN) -
4 * X * X / (E * E *  COS (AN)
*  COS (AN)))
190  IF Y > 160 THEN 250
200  IF (X2 >  = ST) AND (X2 <
ST + C * 8 + 7) AND (Y > 157 -
H * 8) THEN 250
210  IF Y > 0 THEN  HPLOT X2,Y·
 POKE 768,2: POKE 769,200 - Y:
CALL 770
220 X = X + 3
230  IF X2 < 259 THEN 170
240  GOTO 290
250  IF Y > 158 THEN Y = 158
260  HPLOT X2,Y TO X2,Y - 8: FO
R X = 1 TO 12:W =  PEEK ( - 163
36): NEXT
280  IF X2 >  = B AND X2 < B +
7 THEN 300
290  FOR I = 1 TO 1000: NEXT :
```

```
IF D < 10 THEN 80
300  FOR I = 1 TO 1500: NEXT :
HOME : TEXT : IF D = 10 THEN  P
RINT : PRINT "TUDO FOI INUTIL!"
: GOTO 310
305  PRINT : PRINT "BOM TIRO!":
 PRINT "VOCE ACERTOU EM ";D;" D
ISPAROS!"
310  FOR I = 1 TO 4000: NEXT :
PRINT : PRINT "OUTRA VEZ...": F
OR I = 1 TO 4000: NEXT : GOTO 7
0
320  FOR X = 0 TO C: FOR Y = 0
TO H: HPLOT ST + X * 8,158 - Y
* 8
330  NEXT : NEXT
340  HPLOT A,158 TO A + 5,158:
HPLOT A + 3,158 TO A + 3,150: H
PLOT B,158 TO B + 5,158: HPLOT
B + 3,158 TO B + 3,150: RETURN

5000  DATA  172,1,3,174,1,3,169
,4,32,168,252,173,48,192,232,20
8,253,136,208,239,206,0,3,208,2
31,96
```

O programa solicita que você especifique a velocidade e o ângulo de lançamento para executar um disparo de um ponto próximo ao canto inferior esquerdo da tela até um alvo situado junto ao canto inferior direito. Dois fatores dificultam o jogo: a determinação aleatória da distância entre o ponto de lançamento e o alvo e a presença de uma barreira de tamanho variável entre os dois pontos. Qualquer trajetória que se escolha deve ser alta o bastante para transpor essa barreira.

AVALIAÇÃO DA DISTÂNCIA

Conhecemos só as posições do atirador, do obstáculo e do alvo. No entanto, podemos estimar corretamente a velocidade e o ângulo necessários para produzir uma curva que leva o projétil acima do obstáculo até atingir o alvo. Com um pouco de prática, você verá que é possível obter grande proporção de acertos em uma série de tiros.

Mas, em geral, as coisas não são tão fáceis. Muitas vezes temos apenas uma idéia aproximada da distância até o alvo, e devemos calcular a velocidade e ângulo de tiro com este dado impreciso. Analisando se o tiro caiu para cá ou para lá do alvo, avaliações mais precisas vão sendo feitas, até que se consiga sucesso. O programa seguinte mostra esse processo. Depois de gravar o programa anterior, apague-o e digite estas linhas:

```
10 CLS
20 INPUT "VELOCIDADE INICIAL
(1-10000 m/s)",sp
30 IF sp<1 OR sp>10000 THEN
GOTO 20
40 INPUT AT 4,0;"ANGULO (1-90
graus)",a
50 IF a<1 OR a>=90 THEN  GOTO
 40
60 LET a=a*(PI/180)
70 LET r=sp*sp*SIN (2*a)/10
80 PRINT AT 10,3;"O ALCANCE F
OI DE ";INT (r+.5);" metros"
90 PRINT AT 20,1;"QUALQUER TE
CLA PARA RECOMECAR   (0 SAI)"
100 PAUSE 0: LET a$=INKEY$: IF
a$="" THEN  GOTO 100
110 IF a$<>"0" THEN  GOTO 10
```



```
10 CLS
20 INPUT" VELOC. INICIAL (1-100
00 M/S)  ";SP
30 IF SP<1 OR SP>10000 THEN 10
40 INPUT" ANGULO (1-90 GRAUS)";
A
50 IF A<1 OR A>90 THEN 40
60 A=A*ATN(1)/45
70 R=SP*SP*SIN(2*A)/10
80 PRINT:PRINT" O ALCANCE FOI D
E";INT(R+.5);"METROS"
90 PRINT:PRINT" QUALQUER TECLA
PARA RECOMECAR   '0' SAI"
100 A$=INKEY$:IF A$="" THEN 100
110 IF A$="0" THEN CLS:END ELSE
 10
```



```
10 CLS
20 INPUT"VELOCIDADE DE LANÇAMEN
TO (1-10000 M/S)";SP
30 IF SP<1 OR SP>10000 THEN 10
40 INPUT"ANGULO DE LANÇAMENTO (
1-90 GRAUS)";A
50 IF A<1 OR A>89 THEN 40
60 A=A*ATN(1)/45
70 R=SP*SP*SIN(2*A)/10
80 PRINT:PRINT"A DISTANCIA ALCA
NÇADA E";INT(R+.5);"METROS"
90 PRINT:PRINT"PRESSIONE <0> PA
RA TERMINAR OU QUALQUER      OUT
RA TECLA PARA CONTINUAR";
100 A$=INKEY$:IF A$="" THEN 100
110 IF A$="0" THEN CLS:END ELSE
 10
```



```
10  HOME
20  INPUT "VELOCIDADE DE LANCAM
ENTO (1-1000 M/S)?  ";SP
30  IF SP < 1 OR SP > 10000 THE
N 10
40  INPUT "ANGULO DE LANCAMENTO
 (1 90 GRAUS)";A
50  IF A < 1 OR A > 89 THEN 40
60 A = A *  ATN (1) / 45
70 R = SP * SP *  SIN (2 * A) /
 10
80  PRINT : PRINT "A DISTANCIA
ALCANCADA E DE "; INT (R + .5);
" METROS"
90  PRINT : PRINT "TECLE <0> PA
RA TERMINAR OU QUALQUER
OUTRA TECLA PARA CONTINUAR";
100  GET A$
110  IF A$ = "0" THEN  HOME : E
ND
120  GOTO 10
```

O programa pede valores para a velocidade inicial (linha 20) e para o ângulo de lançamento (linha 40). A linha 70 calcula e mostra a distância a que esses dois valores levariam o projétil. A variável **R** corresponde à distância; **SP** refere-se à velocidade e **A**, ao ângulo. O valor aproximado para a aceleração da gravidade próximo à superfície da Terra é igual a 10.

Ignora-se o efeito da resistência do ar mas, em experimentos reais, ele deveria ser considerado. A maior velocidade de lançamento permitida é 10000 m/s. A velocidade de 2000 m/s foi usada em lançamentos de objetos a grandes altitudes, para pesquisar o espaço. Os objetos atingem 180 km de altitude quando lançados a um ângulo de 90°. Mas, sem contar a resistência do ar, chegam a 250 km.

CIRCUNDANDO A TERRA

Se um objeto é lançado em um ângulo que o leva muito longe da superfície da Terra, o efeito da fricção do ar diminui. Porém, quando se pretende que ele atinja uma grande distância do ponto de lançamento, é preciso levar em conta a curvatura da Terra.

Um dos primeiros a considerar esta questão foi o cientista inglês Isaac Newton. Ele imaginou um poderoso canhão que lançasse seus projéteis do alto de uma montanha suficientemente grande para que seu pico ficasse fora da atmosfera terrestre. Tiros dados a velocidades crescentes levariam a distâncias também crescentes se a Terra fosse plana. Mas, como ela é curva, a superfície se afasta do ponto de lançamento, fazendo com que o projétil percorra distâncias ainda maiores.

Eventualmente, argumentava Newton, um lançamento poderia ter uma velocidade inicial tão alta que o projétil nunca atingiria o solo, mas poderia, em termos teóricos, atingir o pesquisador bem na nuca. À medida que o projétil caísse em direção ao solo, este "cairia" sob ele (visto que a Terra é redonda). Dessa maneira, o projétil ficaria para sempre em queda livre — ou seja, estaria em órbita.

Uma vez que um objeto escapa da gravidade de um planeta, sua velocidade e distância determinam um tipo de órbita — circular ou elíptica.

Para que se possa prever a trajetória e fazer medições relacionadas a um planeta ou satélite, sua órbita precisa ser conhecida, ou seja, devemos saber a forma exata da elipse. O grau de "achatamento" da elipse é sua excentricidade (**E**), que corresponde à proporção entre seu comprimento e sua largura. Quando **E** é igual a 1, temos uma elipse tão larga quanto comprida, ou seja, um círculo.

Digite e execute este programa para ver o efeito da variação de **E** entre valores menores e maiores que 1:

```
10 CLS
30 INPUT "EXCENTRICIDADE (.1
a 1.9)",e
40 IF e<.1 OR e>1.9 THEN
GOTO 30
50 PLOT 127,87+e*40
60 FOR a=0 TO 2*PI+.2 STEP .1
70 DRAW 127+(40*SIN a)-PEEK
23677,87+(e*40*COS a)-PEEK
23678
80 NEXT a
90 PRINT AT 20,1;"QUALQUER TE
CLA PARA RECOMECAR   (0 SAI)"
100 PAUSE 0· LET a$=INKEY$·
IF a$="" THEN  GOTO 100
110 IF a$<>"0" THEN  GOTO 10
```



```
10 PMODE 4,1:PCLS
30 CLS:INPUT"EXCENTRICIDADE (.1
 A 1.9) ";E
40 IF E<.1 OR E>1.9 THEN 30
50 SCREEN 1,1
70 CIRCLE(128,96),48,5,E
100 A$=INKEY$:IF A$="" THEN 100
110 IF A$="0" THEN CLS:END ELSE
 30
```



```
30 CLS·INPUT"EXCENTRICIDADE (0 "
1 A 1.9)";E
40 IF E< 1 OR E>1 9 THEN 10
50 SCREEN 2·COLOR 15,4,4
70 CIRCLE(128,96),48,15,,,E
100 A$=INKEY$·IF A$="" THEN 100
110 IF A$="0" THEN CLS·END ELSE
 30
```



```
10  TEXT
30  HOME · INPUT "EXCENTRICIDAD
E (0.1 A 1.9) ";E
40  IF E < .1 OR E > 1.9 THEN 3
0
50  HGR2 · HCOLOR= 3
60 X = 130·Y = 90
70  FOR A = 0 TO 2 * 3.1416 STE
P  05
80  HPLOT X + 45 *  SIN (A),Y -
 (E * 45 *  COS (A))
90  NEXT
100  GET AS
110  IF AS = "0" THEN  TEXT : H
OME : END
120  GOTO 10
```

O programa gira entre as linhas 30 e 110 para desenhar elipses. A linha 30 determina o valor de **E** de acordo com sua escolha. O TRS-Color e o MSX usam o comando **CIRCLE** (linha 70) para desenhar as curvas. Os outros micros empregam um laço **FOR...NEXT** para desenhar cada ponto.

Digite valores de **E** na faixa indicada na tela. **E** igual a 1 corresponderá a um círculo; **E** menor que 1, a uma elipse achatada em cima e embaixo, e **E** maior que 1, a uma elipse achatada lateralmente. Assim, toda órbita pode ser considerada uma elipse, com um valor particular de **E**.

Uma vez em órbita, um satélite ou uma espaçonave não precisam de propulsão, pois estarão caindo livremente. O uso de foguetes irá movê-los para uma órbita diferente. Quanto menor for o raio da órbita, maior será a velocidade com que o objeto deve se mover. Digite o próximo programa para ver como isso funciona:



S

```
10 CLS : LET gc=0
40 LET r=40: LET rt=10+INT (
RND*60): IF ABS (r-rt)<10
THEN  GOTO 40
50 CIRCLE 127,87,4
60 LET s=.1: LET a=0: LET at=
INT (RND*10)+1: LET f=0
80 LET a=a+s
100 LET at=at+.1*SQR ((40/rt)^
3)
110 IF INKEY$="7" AND r<85
THEN  LET f=f+1: LET r=r+2:
```



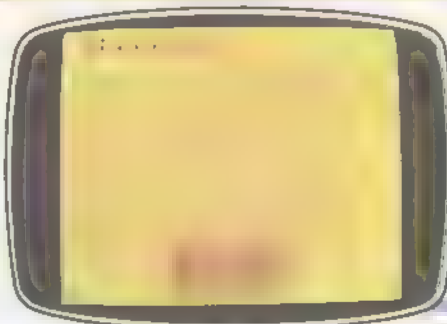
```
LET s=s*SQR (((r+2)/r)^3):
GOTO  130
115 IF INKEY$="6" AND r>8 THEN
 LET f=f+1: LET r=r-2: LET s=s
*SQR (((r+2)/r)^3): GOTO 130
120 FOR p=1 TO 5: NEXT p
130 LET x=r*SIN a: LET y=r*COS
a
140 LET xt=rt*SIN at: LET yt=r
t*COS at
150 PLOT 127+x,87+y
160 SOUND  02,r/2
165 IF oc<>0 THEN  PLOT  OVER
1;127+ox,87+oy
170 PLOT  OVER 1,127+xt,87+yt
LET ox=xt: LET oy=yt: LET oc=1
180 IF ABS (x-xt)>4 OR ABS (y-
yt)>4 THEN  GOTO 80
190 PRINT AT 20,12;f;" PONTOS"
200 GOTO 200
```



```
10 PMODE 4:PCLS:SCREEN 1,1
40 R=30:RT=10+RND(70):IF ABS(R-
RT)<20 THEN 40
50 DRAW"BM128,96S8NUNRNDL"
60 S=.1:A=0:AT=RND(10):F=0
80 A=A+S
95 LT=AT
100 AT=AT+.1*SQR((40/RT)^3)
110 IF PEEK(341)=247 AND R<95 T
HEN F=F+1:R=R+1:S=S*SQR(((R-1)/
R)^3)ELSE IF PEEK(342)=247 AND
R>8 THEN F=F+1:R=R-1:S=S*SQR(((
R+1)/R)^3)ELSE FOR K=1 TO 25:NE
XT
130 X=R*SIN(A):Y=R*COS(A)
140 XT=RT*SIN(AT):YT=RT*COS(AT)
150 PSET(128+X,96-Y,5)
160 SOUND R*2,1
170 PSET(128+XT,96-YT,5)
175 PSET(128+RT*SIN(LT),96-RT*C
OS(LT),0)
180 IF ABS(X-XT)>=4 OR ABS(Y-YT
)>=4 THEN 80
190 CLS:PRINT" VOCE USOU";F;"TO
QUES"
```



```
5 W=RND(-TIME)
10 SCREEN2
40 R=30:RT=INT(RND(1)*70)+11:IF
 ABS(R-RT)<20 THEN 40
50 CIRCLE (128,96),2
60 S=.1:A=0:AT=INT(RND(1)*11)+1
:F=0
80 A=A+S
95 LT=AT
100 AT=AT+.1*SQR((40/RT)^3)
105 FOR I=1 TO 5:A$=INKEY$:IF A
$<>"" THEN I=5
234 NEXT
110 IF A$=CHR$(30) AND R<95 THE
N F=F+1:R=R+1:S=S*SQR(((R-1)/R)
^3)
120 IF A$=CHR$(31) AND R>8 THEN
 F=F+1:R=R-1:S=S*SQR(((R+1)/R)^
3)
```



Um jogo simples usando trajetórias.



Órbita de quatro planetas internos.

```
130 X=R*SIN(A):Y=R*COS(A)
140 XT=RT*SIN(AT):YT=RT*COS(AT)
150 PSET (128+X,96 Y)
170 PSET (128+XT,96-YT)
175 PSET (128+RT*SIN(LT),96-RT*
COS(LT)),0
180 IF ABS(X-XT)>=4 OR ABS(Y-YT
)>=4 THEN 80
190 SCREEN0:PRINT"VOCE USOU";F;
"TOQUES"
```



```
10  HGR2
40 R = 30:RT =  INT ( RND (1) *
 70) + 11: IF  ABS (R - RT) < 2
0 THEN 40
50  HPLOT 126,96 TO 130,96: HPL
OT 128,94 TO 128,98
60 S = .1:A = 0:AS =  INT ( RND
 (1) * 11):F = 0
80 A = A + S
95 LT = AS
100 AS = AS + .1 *  SQR ((40 /
RT)  ^ 3)
105  POKE  - 16368,0
110  IF  PEEK ( - 16384) = 181
AND R < 95 THEN F = F + 1:R = R
 + 1:S = S *  SQR (((R - 1) / R
) ^ 3)
120  IF  PEEK ( - 16384) = 182
AND R > 8 THEN F = F + 1:R = R
- 1:S = S *  SQR (((R + 1) / R)
 ^ 3)
125  POKE  - 16368,0
130 X = R *  SIN (A):Y = R *  C
OS (A)
140 XT = RT *  SIN (AS):YT = RT
 *  COS (AS)
150  HPLOT 128 + X,96 - Y
170  HPLOT 128 + XT,96 - YT
175  HCOLOR= 0: HPLOT 128 + RT
*  SIN (LT),96 - RT *  COS (LT)
: HCOLOR= 3
180  IF  ABS (X - XT) >  = 4 OR
  ABS (Y - YT) >  = 4 THEN 80
190  TEXT : HOME : PRINT "VOCE
USOU ";F;" TOQUES"
```

MANOBRAS NO ESPAÇO

Executando o programa, você verá um satélite-alvo e a trajetória de uma nave em órbita. Tente igualar as duas órbitas usando as setas para cima e para baixo (elas correspondem às teclas 6 e 7 no Spectrum e no Apple).

A linha 40 determina o raio R da órbita da nave como 200; o raio do alvo é definido ao acaso. A linha 60 especifica as variáveis para as posições iniciais. O ponto essencial do programa está na linha 100, que usa uma importante lei da física: o quadrado do tempo para percorrer uma órbita completa dividido pelo cubo do raio é uma constante. Por essa razão, o SQR e a potência 3 estão na linha 100.

Faça as manobras utilizando as teclas citadas para aumentar ou diminuir a órbita. Lembre-se de que quanto menor a órbita, maior a rapidez.

MOVIMENTO PLANETÁRIO

Na realidade, manobrar de uma órbita para outra é muito mais complicado do que sugere o programa anterior. A dificuldade não se encontra na mudança de uma órbita para outra, mas na localização de um alvo na vastidão do espaço. E, para complicar um pouco mais, é preciso levar em consideração que a gravidade do Sol, da Lua e, também, dos planetas tem efeito sobre o movimento de uma espaçonave.

Na prática, poderosos computadores são as ferramentas dos navegantes do espaço. Eles controlam a velocidade, o tempo e a direção dos foguetes, garantindo que estes se mantenham na trajetória desejada.

Uma vez na órbita correta, a nave cai livremente no campo gravitacional do sistema solar. Ela precisa apenas de pequenas correções de curso no decorrer de meses e até anos de viagem — é quase tão previsível quanto os planetas que circundam o Sol.

O programa seguinte permitirá a observação do movimento dos planetas.

```
10 BORDER 4: CLS : LET gc=0
20 DIM d(9): DIM p(9): DIM i(
9): DIM a(9): DIM b(9): DIM x
(9): DIM y(9)
30 FOR t=1 TO 9: READ d(t),p(
t): NEXT t
40 INPUT "Quantos planetas (1
-9) ?",s
50 IF s<1 OR s>9 THEN  GOTO
 40
60 LET sc=d(s)/325: LET t=p(s
)/75
70 PRINT "Ha um espaco de ";
INT t;" dias"'"entre cada pon
to.": PRINT #1;" <SPACE> PARA
 CONTINUAR"
75 IF INKEY$<>CHR$ 32 THEN
GOTO 75
80 CLS
100 PRINT #1;"PRESSIONE <SPACE
> PARA RECOMECAR"
110 LET n=1: LET m=[illegible]: LET g=PI
/180
120 FOR q=1 TO s
130 LET r=d(q)/sc: LET a=r:
LET b=r: LET e=0: LET p=p(q)/t
140 IF q=1 THEN  LET e=.2: LET
b=a*.98
150 IF q=8 THEN  LET e=.26:
LET b=a*.96
160 LET i(q)=i(q)+360/p
180 LET y=g*i(q): LET x=INT (a
*(COS y-e)): LET y=INT (b*SIN
y)
200 IF gc<>0 THEN PLOT
BRIGHT 0;127+x(q),87+y(q)
210 PLOT  BRIGHT 1;127+a(q)/4,
87+b(q)/4: LET x(q)=a(q)/4:
LET y(q)=b(q)/4
240 LET a(q)=x: LET b(q)=y:
NEXT q
250 LET m=m+t
260 IF INKEY$=CHR$ 32 THEN
RUN
270 LET gc=1: GOTO 120
280 DATA 58,88,108,225,150,365
,228,687
290 DATA 778,4333,1427,10759,
2870,30685
300 DATA 4497,60190,5969,90741
```



```
10 PMODE 4:PCLS
20 DIM D(8),P(8),I(8),A(8),B(8)
30 FOR T=0 TO 8:READ D(T),P(T):
NEXT
40 CLS:INPUT " QUANTOS PLANETAS
 (1 9) ";S
50 IF S<1 OR S>9 THEN 40
60 S=S-1:SC=D(S)/90:T=P(S)/75
70 PRINT:PRINT" HA UM ESPACO DE
";INT(T);"DIAS":PRINT" ENTRE CA
DA PONTO":PRINT:PRINT:PRINT "PR
ESSIONE<ESPACO>PARA CONTINUAR"
75 IF INKEY$<>" " THEN 75
80 SCREEN 1,1
90 CIRCLE(128,96),1,5
110 G=ATN(1)/45
120 FOR Q=0 TO S
130 R=D(Q)/SC:A=R:B=R:E=0:P=P(Q
)/T
140 IF Q=0 THEN E=.2:B=A*.98
150 IF Q=8 THEN E=.26:B=A*.96
155 IF S>5 AND Q<6 THEN 245
160 IF P>3 THEN P=INT(P+.5)
170 I(Q)=I(Q)+360/P
180 Y=G*I(Q):X=INT(A*(COS(Y)-E)
):Y=INT(B*SIN(Y))
200 CIRCLE(128+A(Q),96-B(Q)),1,
0
220 PSET (128+A(Q),96-B(Q),5)
230 CIRCLE(128+X,96-Y),1,5
240 A(Q)=X:B(Q)=Y
245 NEXT
250 M=M+T
260 IF INKEY$=" " THEN RUN
270 GOTO 120
280 DATA 58,88,108,225,150,365,
228,687
290 DATA 778,4333,1427,10759,28
70,30685
300 DATA 4497,60190,5969,90741
```



```
10 CLS
20 DIM D(8),P(8),I(8),A(8),B(8)
30 FOR T=0 TO 8:READ D(T),P(T):
NEXT
40 INPUT"QUANTOS PLANETAS (1-9)
";S
50 IF S<1 OR S>9 THEN 40
60 S=S-1:SC=D(S)/90:T=P(S)/75
70 PRINT:PRINT"HA UM ATRASO DE"
;INT(T);"DIAS ENTRE CADA PONTO"
:PRINT:PRINT:PRINT"TECLE A BARR
A DE ESPAÇOS PARA CONTINUAR"
75 IF INKEY$<>" " THEN 75
80 SCREEN 2
90 CIRCLE (128,96),1,15
100 G=ATN(1)/45
120 FOR Q=0 TO S
130 R=D(Q)/SC:A=R:B=R:E=0:P=P(Q
)/T
140 IF Q=0 THEN E=.2:B=A*.98
150 IF Q=8 THEN E=.26:B=A*.96
160 IF P>3 THEN P=INT(P+.5)
170 I(Q)=I(Q)+360/P
180 Y=G*I(Q):X=INT(A*(COS(Y)-E)
):Y=INT(B*SIN(Y))
200 CIRCLE(128+A(Q),96-B(Q)),1,
15
220 PSET(128+A(Q),96-B(Q)),5
230 CIRCLE(128+X,96-Y),1,6
240 A(Q)=X:B(Q)=Y
245 NEXT
250 M=M+T
260 IF INKEY$=" " THEN RUN
270 GOTO 120
280 DATA 58,88,108,225,150,365,
228,687
290 DATA 778,4333,1427,10759,28
70,30685
300 DATA 4497,60190,5969,90741
```



```
10  TEXT
20  DIM D(8),P(8),I(8),A(8),B(8
)
30  FOR T = 0 TO 8: READ D(T),P
(T): NEXT
40  HOME : INPUT "QUANTOS PLANE
TAS (1-9) ";S
50  IF S < 1 OR S > 9 THEN 40
60 S = S - 1:SC = D(S) / 90:T =
 P(S) / 75
70  PRINT : PRINT "HA UM ATRASO
 DE "; INT (T);" DIAS ENTRE CAD
A PONTO": PRINT : PRINT "TECLE
A BARRA DE ESPACOS PARA CONTINU
AR"
75  GET A$: IF A$ <  >  CHR$ (3
2) THEN 75
80  HGR2 : HCOLOR= 3
90  HPLOT 128,96
110 G =  ATN (1) / 45
120  FOR Q = 0 TO S
130 R = D(Q) / SC:A = R:B = R:E
 = 0:P = P(Q) / T
140  IF Q = 0 THEN E = .2:B = A
 * .98
150  IF Q = 8 THEN E = .26:B =
A * .96
155  IF S > 5 AND Q < 6 THEN 24
5
160  IF P > 3 THEN P =  INT (P
 + .5)
170 I(Q) = I(Q) + 360 / P
180 Y = G * I(Q):X =  INT (A *
( COS (Y) - E)):Y =  INT (B *
SIN (Y))
200  HCOLOR= 0: HPLOT 129 + A(Q
),96 - B(Q) TO 128 + A(Q),95 -
B(Q) TO 127 + A(Q),96 - B(Q) TO
 128 + A(Q),97 - B(Q)
210  HCOLOR= 3: HPLOT 128 + A(Q
),96 - B(Q)
220  HPLOT 129 + X,96 - Y TO 12
8 + X,95 - Y TO 127 + X,96 - Y
TO 128 + X,97 - Y
240 A(Q) = X:B(Q) = Y
245  NEXT
250 M = M + T
260  POKE  - 16368,0
265  IF  PEEK ( - 16384) > 128
THEN  RUN
270  GOTO 120
280  DATA  55,88,108,225,150,36
5,228,687
290  DATA  778,4333,1427,10759,
2870,30685
300  DATA  4497,60190,5969,9074
1
```

Ao executar o programa, você deve escolher o número de planetas que deseja ver. Quanto maior o número deles, mais complicada a figura. Para identificar os planetas, lembre-se de que Mercúrio está mais perto do Sol, seguido por Vênus, Terra, Marte, Júpiter, Saturno, Urano, Netuno e Plutão.

Execute o programa com diferentes valores. Note que as órbitas de dois planetas — Mercúrio e Plutão — são elípticas. Na verdade, as outras também o são, mas têm uma excentricidade tão pequena que parecem circulares.

AVALANCHE: EFEITOS SONOROS

Vamos ouvir um som antes de colocar nosso personagem em cena. Que tal um grande sucesso de 1560? Sente-se diante de seu micro e prepare uma abertura musical digna de Avalanche.

Um jogo de ação não seria completo sem uma abertura musical. Nosso videogame toca a melocia *Greensleeves*, no tom e ritmo corretos. Talvez você critique nossa escolha, por julgá-la pouco adequada a um videogame. Contudo, de acordo com a lenda, Henrique VIII escreveu essa melodia para sua esposa Ana Bolena. Mais tarde, ela teve um fim bastante trágico, assim como será o de Willie, se cometer qualquer descuido no jogo. Além disso, não teremos que pagar direitos autorais — o que é muito importante...



A rotina Assembly listada a seguir utiliza a sub-rotina da ROM que controla o comando **SOUND**, o que facilita a execução das notas musicais.

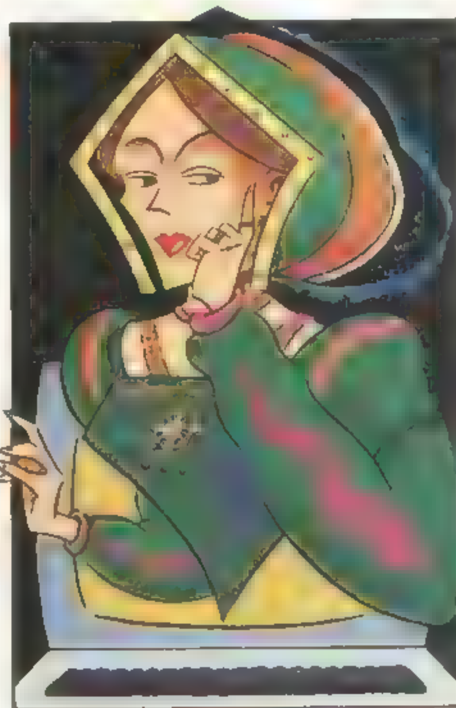
```
 10 REM org 60000
 20 REM ld ix,57359
 30 REM msk ld b,19
 40 REM tune push bc
 50 REM ld d,(ix+1)
 60 REM ld e,(ix+0)
 70 REM ld h,(ix+3)
 80 REM ld l,(ix+2)
 90 REM push ix
100 REM call 949
110 REM pop ix
120 REM ld de,4
130 REM add ix,de
140 REM pop bc
150 REM djnz tune
160 REM ret
```

Podemos escolher qualquer melodia, contanto que forneçamos a essa rotina as informações necessárias sobre o que vai tocar. O programa BASIC que apresentamos a seguir coloca na memória os dados que o computador precisa para executar *Greensleeves*. As notas, com suas respectivas durações, formarão uma espécie de tabela, colocada logo abaixo da tabela ASCII que contém os textos do título e das instruções.

```
 5 CLEAR 57358
10 FOR n=57359 TO 57434 STEP
2: READ a· POKE n+1,INT (a/
256). POKE n,a-(256*INT (a/
256)): NEXT n
20 DATA 98,1460,233,1223,131,
1086,220,964,78,908,147,964,
261,1086,110.1297,131,1642,49
.1460,110,1297,233,1223,98,
1460,147,1460,44,1642,98,1460
,220,1297,92,1548.220.1959
```

COMO FUNCIONA

A rotina começa colocando o endereço 57359 no registro IX. Como registro-índice, IX vai servir de apontador e seu valor inicial corresponde ao início da tabela de notas.



Em seguida, 19 é colocado em B, registro que servirá como contador. Seu conteúdo inicial corresponde ao número de notas que serão executadas (o trecho de *Greensleeves* que vamos tocar tem dezenove notas). Esse valor é guardado temporariamente na pilha, pelo comando **push bc**. O registro C precisa ser guardado junto com o B porque não existe um comando que guarde registros de oito bits na pilha. As instruções **push** e **pop** só aceitam pares de registros como operandos.

Mas por que guardar o valor de B e C na pilha, se nenhum dos dois registros será usado até que a instrução **pop bc** recupere seu conteúdo original? Esta é uma boa pergunta. Enquanto o valor de BC está na pilha, o programa chama uma sub-rotina da ROM, que pode muito bem usar o registro B. Embora uma sub-rotina possa alterar o valor de qualquer registro, ela sempre deixa a pilha intacta. Assim, sempre que chamamos uma sub-rotina que não conhecemos bem, devemos guardar o conteúdo dos registros importantes na pilha.

O primeiro número da tabela é colocado em DE; o segundo, em HL. A operação, diferentemente do usual, emprega comandos com endereçamento indexado, de modo que IX é usado como apontador. Os registros têm que ser carregados um de cada vez, já que não há comandos que carreguem pares de registros nesse tipo de endereçamento. Se consultarmos a listagem do programa BASIC, veremos que cada número ocupa dois bytes, ainda que muitos sejam menores que 256.

Os números formam pares. O primeiro valor determina a duração da nota — na realidade, ele especifica o número de ciclos gerados pela rotina do comando **SOUND**. O número corresponde, então, à freqüência da nota multiplicada pela sua duração em segundos. O segundo número determina a tonalidade da nota. Para calcular seu valor, podemos usar a fórmula 437500/f-30125, onde **f** é a freqüência da nota.

A rotina coloca o conteúdo de IX na pilha, para preservar o valor do apontador, e, em seguida, chama a sub-rotina da ROM. A sub-rotina do comando **SOUND** é chamada pela instrução **call 949**. Ela usa os valores de HL e DE para produzir o som. A instrução **pop ix** recupera da pilha a posição do apontador na tabela de notas; o ponteiro avança um par de notas, com a colocação de 4 em DE (**ld de, 4**) e a soma desse valor a IX (**add ix,de**). O emprego de **add** é necessário, já que não existe uma instrução especial para somar valores ao índice IX. Poderíamos usar quatro instruções **inc ix**, mas o processo seria mais demorado.

O valor original do contador volta para BC. A instrução **djnz** subtrai uma unidade de B e, se este não foi ainda re-

- A ESCOLHA DA MÚSICA
- CONTAGEM DAS NOTAS
- CÁLCULO DAS FREQÜÊNCIAS
- ROTINA PRINCIPAL E SUB-ROTINAS
- ARMAZENAGEM DA MÚSICA
- COMO MODIFICAR A MELODIA
- COMO SINCRONIZAR O SOM E A AÇÃO DO JOGO



duzido a zero, ela faz o processador voltar para emitir mais uma nota. Após dezenove notas, o processador deixa o laço e encontra **ret**.

Para testar a sub-rotina, utilize **RAND USR 60000**, mas não se esqueça de que a tabela de notas deve estar registrada na memória.

TOQUE SUA PRÓPRIA MELODIA

Mesmo depois de toda aquela conversa sobre Henrique VIII, talvez você queira ouvir outra música. Embora não seja aconselhável fazer a substituição dentro do nosso jogo, você pode usar a rotina separadamente.

Escolha a música de sua preferência e traduza-a nos parâmetros que a rotina utiliza para emitir notas musicais. Lembre-se, contudo, de que melodias diferentes requerem um outro valor para o contador B, que determina o número de notas executadas.



O TRS-Color executa a melodia *Greensleeves* por intermédio da rotina Assembly, que é dada a seguir. Não se esqueça de aumentar a memória disponível com **POKE 25,6:POKE 26,1:NEW** e também de proteger uma área para o programa em código.

```
10  ORG 30000
20  LDX #MUSIC
30  STX MUPOINT
40  LDA #19
50  PSHS A
60  LDA $FF01
70  ANDA #247
80  STA $FF01
90  LDA $FF03
100  ANDA #247
110  STA $FF03
120  LDA $FF23
130  ORA #8
140  STA $FF23
150  MAIN LDU MUPOINT
160  ORCC #$50
170  PULU A,X
180  CMPU #MUSIC+57
190  BLO MONE
200  LDU #MUSIC
210  MONE STU MUPOINT
220  PSHS X
230  LDB #252
240  MTWO STB $FF20
250  MTHR LEAX -1,X
260  BNE MTHR
270  LDX ,S
280  CLR $FF20
290  MFOU LEAX -1,X
300  BNE MFOU
310  LDX ,S
320  DECA
330  BNE MTWO
340  LEAS 2,S
350  DEC ,S
360  BNE MAIN
370  ANDCC #$AF
380  PULS A,PC
390  MUPOINT FDB $758A
400  MUSIC FCB 98,0,189,233
410  FCB 0,158,131,0
420  FCB 141,220,0,125
430  FCB 78,0,118,147
440  FCB 0,125,255,0
450  FCB 141,110,0,168
460  FCB 131,0,212,49
470  FCB 0,189,110,0
480  FCB 168,233,0,158
490  FCB 98,0,189,147
500  FCB 0,189,44,0
510  FCB 212,98,0,189
520  FCB 220,0,168,92
530  FCB 0,200,220,0,252
```

O programa tem duas entradas: uma fica em 30000 e outra em 30008 (linha 50). Se a rotina for chamada pelo primeiro endereço, executará a melodia inteira; se for chamada pelo segundo, executará apenas uma nota. Esta última opção permite que se toque a melodia sem interromper a ação do jogo. Podemos tocar uma nota, fazer alguma outra coisa na tela, tocar outra nota, e assim por diante.

ARMAZENE A MELODIA

O valor do rótulo **MUSIC** é colocado no byte rotulado **MUPOINT**. Embora na listagem — e, posteriormente, na memória — o valor de **MUSIC** já esteja nesse byte, usa-se **MUPOINT** para guardar a posição da última nota executada, o que é útil quando tocamos a melodia uma nota de cada vez.

O comando **LDA #19** coloca no acumulador o número de notas do tre-

cho de *Greensleeves* que vamos executar — dezenove. Se você não quiser tocar a melodia inteira de uma vez, coloque no acumulador o número de notas que pretende ouvir. Assim, optando pela execução da música nota por nota, você deverá colocar 1 em A.

O conteúdo de A é levado para a pilha da máquina por **PSHS A**. Trata-se, porém, apenas de uma armazenagem temporária. Mais tarde, o programa precisará do número de notas.

As nove instruções que se seguem viabilizam a produção de sons, colocando os valores apropriados no chip de som do TRS-Color.

A ROTINA PRINCIPAL

MUPOINT aponta para a próxima nota a ser executada, de modo que o endereço do próximo byte da tabela de notas é colocado no apontador da pilha do usuário pelo comando **LDU MUPOINT**. Isto efetivamente transforma a tabela de notas na pilha do usuário.

A instrução **ORCC #$50** impede — "desabilita", no jargão da informática — interrupções, mascarando o conteúdo do registro indicador de status. Quaisquer que sejam os valores dos bits 4 e 6 desse registro, eles passam a ser 1 quando fazemos a operação "ou" entre seu conteúdo e 50 em hexadecimal (80 em decimal).

Como U está apontando para a tabela de notas, a instrução **PULU A,X** traz para dentro de A o próximo byte da tabela, e para dentro de X os dois bytes seguintes. Além disso, também soma 3 ao registro U, que passa a apontar para o próximo byte.

Os dados necessários para o som vêm em grupos de três bytes. O primeiro corresponde ao número de vezes que a nota será tocada. Os dois seguintes determinam a duração que influi na frequência da nota.

Em seguida, o registro U é comparado com **#MUSIC+57**, o endereço final da tabela de notas. O comando **BLO** provoca um desvio se o valor de U for menor, pois, nesse caso, ainda não chegamos ao fim da tabela e a próxima instrução será ignorada. Ela colocaria o endereço inicial da tabela de volta no apontador e a música poderia, então, ser tocada novamente.

De qualquer maneira, o valor do apontador volta para **MUPOINT**, para que o processador saiba qual é a próxima nota. A duração da nota, que se encontra em X, é colocada temporariamente na pilha da máquina.

As instruções **LDB #252** e **STA $FF20**, referentes ao conversor analógico-digital, preparam-se para usar o alto-falante da TV. **LEAX −1,X** diminui o valor de X, e **BNE** faz o processador repetir a instrução anterior, até que X seja zero.

LDX ,S coloca o valor da pilha em X, funcionando como um **PULS X**, sem somar nada, no entanto, ao apontador da pilha. **CLR $FF20** desliga o alto-falante da TV e a mesma rotina de contagem em X é repetida.

O valor da pilha retorna a X e o número de vezes que a nota vai ser tocada é diminuído em uma unidade — **DECA**. Outra instrução **BNE** faz o processador repetir o laço **MTWO**, até que A seja reduzido a zero.

A instrução **LEAS 2,S** limpa a pilha e **DEC ,S** subtrai 1 do conteúdo do próximo byte da pilha da máquina — número de notas, colocado na pilha no início. **BNE MAIN** faz o processador voltar e executar a próxima nota, caso o número de notas não tenha sido reduzido a zero.

ANDCC #AF permite novamente — ou seja, "reabilita" — a ocorrência de interrupções, "desmascarando" o registro de status. Finalmente, **PULS A,PC** retira dois valores da pilha da máquina, colocando em A o valor da nota — reduzido a zero — e em **PC**, os próximos dois bytes. Eles correspondem ao endereço de retorno da sub-rotina, colocado ali quando a rotina foi chamada. O procedimento, nesse caso, é equivalente a um comando **RTS**.



Para dotarmos nosso videogame de uma trilha sonora, precisaremos utilizar o microprocessador especial que o MSX possui para produzir sons: o PSG — *Programable Sound Generator*. Apresentamos, no artigo publicado à página



Qual a diferença entre os comandos *outi* e *otir* no MSX?

No artigo da página 213, vimos como empregar o comando de ação em bloco **otir**. Aqui, utilizamos um comando muito parecido: **outi**

Tanto **otir** como **outi** são usados quando se quer transferir dados que estão em uma tabela na memória RAM, através de uma porta de saída. Esses dois comandos agem sobre os registros HL, B e C. HL é usado como apontador, B como contador e C contém o número da porta de saída. Assim, antes da primeira execução do comando, o par HL deve conter o endereço inicial da tabela, e o registro B, o número de elementos da mesma.

Ambos somam uma unidade ao conteúdo de HL e subtraem o mesmo de B. Mas há uma diferença entre eles: o comando **otir** repete esta operação muitas vezes, transferindo vários caracteres até que B seja reduzido a zero; **outi**, por sua vez, só transfere um número a cada execução.

O comando **otir** é, por si mesmo, um laço completo, utilizando B como contador e fazendo sozinho o teste do indicador de zeros. Mostra-se, portanto, mais adequado quando queremos transferir dados a grande velocidade. No artigo anterior, aqui mencionado, ele foi utilizado para transferir um banco de caracteres para a memória de vídeo.

Já a instrução **outi** deve fazer parte de um laço. Ela vai diminuindo o valor de B, mas não é capaz de descobrir por si mesma que este foi reduzido a zero. Para isso, o programador precisa incluir um teste do tipo **jr nz** O emprego da instrução **outi** é, assim, mais indicado quando precisamos interromper a transferência dos dados a todo instante para a realização de algum tipo de processamento. Em nosso caso, jamais poderíamos transferir valores para o PSG em alta velocidade — obteríamos barulho, e não música.

A transferência foi feita numa velocidade irregular, sendo interrompida por pausas cuja duração é variável A execução dessas pausas e a constante mudança nos números dos registros que estavam sendo alterados — enviados por out **160,a** — correspondem à realização de algum tipo de processamento.

168, uma introdução ao funcionamento desse dispositivo.

O PSG possui catorze registros. Os valores ali colocados determinam os diversos parâmetros do som produzido. Podemos controlar a tonalidade, o volume e sua evolução ao longo do tempo (envoltória ou *envelope*). Temos à nossa disposição três canais, que funcionam simultaneamente, emitindo cada qual um som diferente. Todos eles podem tocar notas musicais e, também, produzir ruídos.

As rotinas Assembly deste artigo executam *Greensleeves* em duas vozes — isto é, são usados dois canais para produzir acordes. Para que o computador saiba que notas tocar e com qual duração, um programa BASIC colocará uma tabela de notas na memória.

CONDIÇÕES INICIAIS DO PSG

A rotina listada a seguir prepara o PSG para tocar a música, colocando valores lidos na tabela dentro dos registros apropriados. Use nosso Assembler para montá-la na memória. Não se esqueça de modificar o comando **CLEAR** da linha 5000, para proteger a memória acima de **&HCFFF**. As rotinas em código devem ser montadas primeiro. Em seguida, pode-se apagar o Assembler e executar o programa BASIC que aparece no final do artigo.

```
10    org -12216
20    ld hl,-14500
30    ld b,6
40    init ld c,160
50    outi
60    ld c,161
70    outi
80    jr nz,init
90    ret
100    end
```

Dispomos de duas alternativas para modificar o valor de um registro do PSG em BASIC. Podemos utilizar tanto o comando **SOUND R,V**— que coloca no registro **R** o valor **V**— quanto a instrução **OUT**.

O microprocessador PSG é considerado um periférico da unidade de processamento central, de maneira que existem dois endereços da área de comunicação — **I/O** — que funcionam como portas de entrada e saída. O valor enviado pela porta 160 indica o registro que será alterado, enquanto a porta 161 corresponde ao destino do novo valor do registro. Dessa maneira, o par de instruções **OUT 160,R:OUT 161,V** equivale ao comando **SOUND R,V**.

Como vimos no artigo da página 213, existe, em linguagem de máquina, uma instrução **out** que funciona de modo parecido ao comando BASIC de mesmo nome. Contudo, como vamos transferir valores de uma tabela para o PSG, usaremos um comando de ação em bloco.

Nossa rotina utiliza o comando **outi**, cujas características devemos entender antes de passarmos a examinar o funcionamento do programa. Esse comando envia o valor contido no endereço apontado pelo par HL através da porta que tem seu número no registro C. Em seguida, o conteúdo do par HL aumenta em uma unidade e o conteúdo de B diminui também em uma unidade.

O comando **outi** é útil para transferir dados de uma tabela, um a um, através de uma porta. Antes de sua primeira aparição, devemos colocar em HL o endereço inicial da tabela, em C, o número da porta, e em B, o número de vezes que queremos repetir o processo. Cada vez que **outi** é executado, HL passa a apontar para o endereço seguinte da tabela. B é diminuído e, quando for reduzido a zero, o indicador de zero será estabelecido.

A primeira instrução determina o endereço inicial da rotina, −12216. Em seguida, os registros HL, B e C recebem os parâmetros da instrução **outi**. O endereço inicial da tabela de notas é colocado em HL por **ld hl,−14450**. Como são necessários três pares de valores para ativar o PSG, a linha **ld b,6** coloca 6 em B. A linha rotulada com **init** coloca em C o número da porta de saída que determina o registro.

Quando se executa a instrução **outi** pela primeira vez, o valor que marca o início da tabela, apontado por HL, é enviado pela porta 160 — conteúdo de C. Se olharmos o programa BASIC no final do artigo, veremos que foi enviado o número 7. Soma-se uma unidade ao par HL, que aponta, então, para o próximo número da tabela, e subtrai-se uma unidade de B.

A instrução **ld c,161** faz com que o valor enviado por **outi** na linha seguinte se dirija à porta 161. Se consultarmos novamente a listagem BASIC, veremos que foi enviado o número 24. O valor de HL aumenta novamente e o de B diminui. Como B ainda não foi reduzido a zero, o processador retorna ao rótulo **init**.

O laço entre as linhas 40 e 80 repete-se três vezes. A instrução **outi** é executada seis vezes, de forma que são enviados três pares de números. Subtrai-se de B uma unidade seis vezes. Na sexta vez, a condição de **jr nz,init** não é satisfeita, e a instrução **ret** provoca o retorno da sub-rotina.

Cada volta desse laço equivale a um comando **SOUND**. O primeiro valor enviado é o registro, e o segundo, seu novo conteúdo. Mas que papel desempenha cada registro?

A primeira volta do laço coloca 24 no registro 7 do PSG. Esse registro seleciona o modo de utilização dos canais de som. Cada um dos seus seis primeiros bits determina se um canal vai produzir som, ruído ou se vai permanecer em silêncio. Quando valem 1, os três primeiros bits ativam os canais A, B e C, respectivamente, para a produção de ruído. Os três bits seguintes ativam os canais para a produção de sons musicais. Como 24 é 00011000 em binário, ele ativa a produção de notas musicais em A e B.

Na segunda volta do laço, o valor 15 vai para o registro 8 do PSG. Esse registro controla o volume do canal A — 15 é o volume máximo.

Na terceira e última volta, o registro 9, que controla o volume do canal B do PSG, recebe o valor 15.

TOCANDO UM ACORDE

Depois de ativar o PSG, podemos começar a tocar nossa música. *Greensleeves* compõe-se de acordes de duas notas, cada uma emitida por um dos canais ativados. As notas serão obtidas na mesma tabela, que especifica também qual é a duração do acorde formado por cada par delas. A rotina listada a seguir executa um acorde. Monte-a e grave-a separadamente.

```
10    org -12200
20    ld hl,-14494
30    ld b,4
40    ld c,161
50    ld a,0
60    tune out 160,a
70    inc a
80    outi
90    jr nz,tune
100    ret
110    end
```

Os acordes são compostos por notas emitidas simultaneamente pelos canais A e B. Registros do PSG controlam a tonalidade de cada nota. Os registros 0 e 1 controlam a tonalidade do canal A, enquanto 2 e 3, a do canal B. Os registros 0 e 2 correspondem aos bytes menos significativos; 1 e 3, aos mais significativos. Para emitir um acorde precisaremos, então, modificar o conteúdo de quatro registros do PSG, usando novamente a instrução **outi**. Por isso, a rotina começa acertando os valores de HL, B e C, depois de estabelecido o endereço inicial.

A instrução **ld hl, – 14494** coloca em HL o endereço da primeira nota da tabela. Como emitiremos quatro notas, o número quatro é colocado em B. O número da porta 161 — utilizada para enviar novos valores de registros ao PSG — é colocado em C.

Nesta rotina, o registro A conterá o número do registro do PSG a ser alterado; assim, seu valor inicial será 0 — **ld a,0**

O comando **out 160,a** envia o conteúdo de A pela porta 160, indicando ao PSG o registro que será alterado. Em seguida, A aumenta em uma unidade. A instrução **outi** envia, então, através da porta 161, o próximo valor da tabela, apontado por HL. HL aumenta em uma unidade e B diminui o mesmo tanto. Enquanto B não for reduzido a zero, a instrução **jr nz,tune** repetirá o laço entre as linhas 60 e 90.

Na primeira volta do laço, o registro A contém 0, de maneira que o registro 0 do PSG recebe o valor da tabela apontado por HL. A cada volta, A aumenta em uma unidade e os registros 1, 2 e 3 do PSG recebem os próximos valores da tabela de notas.

Na quarta vez que **outi** está sendo executado, o valor de B reduz-se a zero. A condição de **jr nz, tune** não é mais satisfeita e a instrução **ret** provoca o retorno da sub-rotina.

CONTROLE DA DURAÇÃO DO ACORDE

O som correspondente ao conteúdo dos registros do PSG é emitido continuamente, até que modifiquemos um de seus parâmetros ou desliguemos o PSG. Assim, para controlarmos a duração do som, usamos uma sub-rotina que não faz absolutamente nada por algum tempo, provocando uma pausa. Enquanto durar essa pausa, o acorde será executado; quando ela terminar, poderemos emitir um novo acorde. A duração será determinada por um número da tabela de notas. Eis a rotina:

```
10    org -12183
20    ld b,(hl)
30    ldp ld hl,1000
40    ld de,0
50    ldq dec hl
60    push hl
70    sbc hl,de
80    pop hl
90    jr nz,ldq
100    djnz ldp
110    ret
120    end
```

Quando a rotina de emissão do acorde termina, deixa em HL o endereço do próximo número da tabela de notas. Esse número indica a duração do acorde,

MICRO DICAS

O PROCESSADOR DO TRS-COLOR

Os usuários do TRS-Color costumam sentir muita dificuldade quando fazem suas primeiras experiências com linguagem de máquina. Para eles, aqui vão algumas recomendações

O processador 6809 é um dos mais poderosos chips de oito bits. Sua versatilidade tem, contudo, um preço elevado, pois o grande número de comandos e formas de endereçamento pode confundir o principiante

No 6502 e no Z-80, processadores das outras máquinas, é possível fazer muita coisa interessante com um pequeno conhecimento dos comandos de armazenamento e de controle de laços. Basta saber que valores colocar nas posições adequadas. Programas um pouco maiores exigem o uso da pilha para a armazenagem temporária, mas isto não é um grande problema.

O 6502, processador do Apple e do TK-2000, conta com vários tipos de endereçamento. Mas são tão poucas as instruções disponíveis que essa vantagem deixa de ser significativa.

O 6809 é um precursor dos chips de dezesseis bits. Toda a sua estrutura de endereçamento e armazenagem baseia-se em números de dois bytes. Seus registros comportam dezesseis bits. Ao mesmo tempo, ele é um chip de oito bits. Cria-se, assim, uma infinidade de modos de endereçamentos — surgindo termos como *pós-byte* e *endereço efetivo* —, o que confunde muita gente. São tantos os comandos de ação em bloco, executando tarefas diferentes, que é difícil deduzir suas funções pelo nome.

O que mais confunde o principiante, contudo, é a utilização das pilhas. Há duas pilhas, a do usuário e a da máquina, cada uma com seu apontador. E elas não são usadas apenas para guardar valores temporariamente. O uso inteligente de seus apontadores permite economizar várias linhas de programação, mas certas operações com as pilhas são de deixar qualquer um atordoado. E, pior ainda, é impossível fazer programas sem esses recursos avançados.

A saída está no estudo cuidadoso das instruções explicadas no texto. Na falta de um manual do 6908, liste as características de cada instrução. Outra medida obrigatória é a leitura dos textos introdutórios à programação em código, já publicados em INPUT. Neles o usuário do TRS-Color encontrará conceitos fundamentais sobre o seu processador.

e é colocado em B pela instrução **ld b,(hl)**. O registro B determinará o número de vezes que o laço rotulado **ldp** vai ser executado. Nesse laço, 1000 é colocado no par HL, que diminui uma unidade a cada volta do laço interno **ldq**. Guarda-se HL temporariamente na pilha, para gastar tempo.

A instrução **sbc hl,de** subtrai zero — conteúdo de DE — do conteúdo de HL. Faz-se isto para afetar o indicador de zeros, quando HL for reduzido a zero, pois a instrução **pop HL** não modifica esse sinalizador. Quando HL contém zero, a condição de **jr nz,ldq** não é satisfeita e o processador passa ao laço externo, controlado por B.

A cada volta do laço externo, B diminui. Quando ele se torna zero, **ret** provoca o retorno da sub-rotina.

Podemos controlar o *andamento* da música aumentando ou diminuindo o número de execuções do laço interno. Para isto, basta colocar um número diferente de 1000 em HL, na linha 30.

A ROTINA PRINCIPAL

O trecho de *Greensleeves* que vamos executar tem 88 acordes. As rotinas que mostramos até agora ativaram o PSG e tocaram um acorde apenas. Para a execução da melodia toda, precisaremos de um programa principal, que chame ordenadamente as sub-rotinas. Monte a rotina que se segue com seu Assembler, gravando depois os códigos.

```
10    org -12166
20    call -12216
30    ld b,88
40    loop push bc
50    call -12197
60    push hl
70    call -12183
80    pop hl
90    inc hl
100    pop bc
110    djnz loop
120    call -12213
130    ret
140    end
```

A primeira providência do programa principal é chamar a sub-rotina que ativa o PSG, que fica no endereço −12216. Quando o processador retorna, HL aponta para a primeira nota da música — sétimo número da tabela.

A instrução **ld b,88** coloca em B o número de acordes que serão executados. Como a rotina que emite o acorde utilizará o registro B como contador, o conteúdo deste é temporariamente guardado na pilha com **push BC**.

A rotina que emite o acorde é, então, chamada. Note que o endereço de transferência é −12197, e não −12200 (veja listagem). Ignora-se a primeira linha — linha 20 — da rotina do acorde, pois HL já contém o valor −14494, que corresponde ao sétimo número da tabela. Essa linha foi incluída no programa para que a melodia possa ser tocada mais de uma vez, sem precisarmos ativar novamente o PSG.

Quando o processador retorna da sub-rotina, o par de registros HL aponta para o próximo número da tabela. Como HL será usado como contador na sub-rotina de pausa, seu valor também é guardado na pilha com **push hl**.

A instrução **call −12183** chama a sub-rotina que é responsável pela duração do acorde.

Quando a rotina de pausa termina, o conteúdo de HL é recuperado da pilha. Esse valor, porém, está "atrasado", pois aponta para o número da tabela que controla a duração da nota, já usado pela rotina de pausa. Para que HL aponte para a próxima posição da tabela, precisamos somar 1 ao seu conteúdo, usando **inc hl**.

O conteúdo de B — nosso contador de acordes — é recuperado da pilha por **pop bc**. Em seguida, a instrução **djnz loop** subtrai 1 de B e volta para o rótulo **loop** enquanto B não tiver sido reduzido a zero.

O laço entre as linhas 40 e 110 é repetido 88 vezes, uma para cada acorde. As notas são executadas até que sua tonalidade se modifique — sendo, portanto, substituídas por outras notas. Se o programa parasse aqui, o PSG ficaria emitindo o último acorde da canção durante todo o jogo, o que você certamente não deseja.

Para desligar o PSG, a rotina que o ativou no início é chamada novamente, só que com outro endereço de transferência. A instrução **call −12213** ignora a primeira linha da sub-rotina, que colocava o endereço inicial da tabela em HL. Assim, como HL ainda aponta para o próximo número não usado na tabela, a sub-rotina utiliza os últimos seis números. Se procurarmos na listagem BASIC, veremos que estes são 8, 0, 9, 0, 10 e 0. Os registros 8, 9 e 10, que controlam o volume dos três canais, recebem, então, o valor 0, e o PSG é silenciado.

Como toda rotina que se preze, esta termina com **ret**.

É interessante notar a importância do apontador HL na execução da melodia. No decorrer da execução, ele percorre toda a tabela. Para que isso seja possível, a cada volta do laço entre as linhas 40 e 110, seu valor é atualizado quatro vezes pela instrução **outi** e uma vez por **inchl** — o que corresponde à transferência de cinco valores da tabela. Quatro deles vão para o PSG e um vai para o registro B controlar a duração da nota. Como a rotina de pausa utiliza HL, entre a execução das linhas 60 e 80 o apontador da posição atual na tabela de notas fica guardado na pilha.

A TABELA DE NOTAS

O conjunto de rotinas apresentado executa uma melodia em duas vozes. Na realidade, ele pode tocar qualquer melodia desse tipo. Cabe à tabela de notas determinar os acordes que serão incluídos. Para criá-la, apague o Assembler e use este programa BASIC.

```
10 CLEAR 200,-14501
20 FOR I=0 TO 451
30 READ A:POKE -14500+I,A
40 NEXT
1000 DATA 7,24,8,15,9,15
1010 DATA 253,0,253,0,10
1020 DATA 213,0,253,0,30
1030 DATA 189,0,213,0,10
1040 DATA 169,0,213,0,15
1050 DATA 159,0,213,0,5
1060 DATA 169,0,28,1,10
```

ERRATA
ASSEMBLER DO TRS-COLOR

Para que seu programa Assembler funcione a contento, você precisará fazer algumas modificações:

```
10 PCLEAR 1:CLEAR 3000:CLS:PRIN
T @233,"INICIALIZANDO":R$=CHR$(
13):POKE 146,1

100 DATA COM,115,3,CWAI,108,1,D
AA,25,,ORA,186,1,TST,125,3,LEAS
,66,3,LEAU,67,3,LEAX,64,3,LEAY,
65,3,MUL,61,,EORA,184,1,ORB,250
,1

1550 P1=1478:P0=P1:P2=0

1560 PRINT @448-P2,K;TAB(6)T$(K
2):P9=P0+LEN(T$(K2))

1565 IF LEN(T$(K2))+P0>1503 THE
N P0=P0-32:P2=P2+32:P1=P1-32:GO
TO 1565

1950 IF X$<>"S" OR BD$<"0" OR B
D$>"G" THEN 1980
```

A alteração na linha 10 é necessária devido aos **POKE** que aumentam a memória disponível. Se você não executar esses comandos antes de ler o Assembler na fita cassete, ele não conseguirá montar o videogame.

As demais modificações ajustam alguns defeitos contidos no programa original.

```
1070 DATA 189,0,28,1,30
1080 DATA 225,0,225,0,10
1090 DATA 28,1,225,0,15
1100 DATA 253,0,225,0,5
1110 DATA 225,0,253,0,10
1120 DATA 213,0,253,0,30
1130 DATA 253,0,253,0,10
1140 DATA 253,0,63,1,15
1150 DATA 12,1,63,1,5
1160 DATA 253,0,63,1,10
1170 DATA 225,0,253,0,17
1180 DATA 225,0,169,0,10
1190 DATA 12,1,225,0,10
1200 DATA 169,0,82,1,20
1210 DATA 253,0,82,1,10
1220 DATA 213,0,253,0,30
1230 DATA 189,0,253,0,10
1240 DATA 169,0,213,0,15
1250 DATA 159,0,213,0,5
1260 DATA 169,0,213,0,10
1270 DATA 189,0,28,1,30
1280 DATA 225,0,28,1,10
1290 DATA 28,1,225,0,15
1300 DATA 253,0,225,0,5
1310 DATA 225,0,225,0,10
1320 DATA 213,0,253,0,30
1330 DATA 225,0,225,0,5
1340 DATA 253,0,213,0,5
1350 DATA 253,0,189,0,5
1360 DATA 12,1,169,0,15
1370 DATA 45,1,169,0,5
1380 DATA 12,1,169,0,10
1390 DATA 253,0,253,0,30
1400 DATA 253,0,126,0,10
1410 DATA 253,0,169,0,10
1420 DATA 253,0,253,0,30
1430 DATA 142,0,213,0,30
1440 DATA 142,0,189,0,5
1450 DATA 142,0,169,0,5
1460 DATA 142,0,169,0,15
1470 DATA 159,0,189,0,5
1480 DATA 169,0,213,0,10
1490 DATA 189,0,28,1,30
1500 DATA 189,0,253,0,5
1510 DATA 225,0,225,0,10
1520 DATA 28,1,225,0,15
1530 DATA 253,0,253,0,5
1540 DATA 225,0,28,1,10
1550 DATA 213,0,253,0,30
1560 DATA 213,0,225,0,5
1570 DATA 253,0,213,0,10
1580 DATA 253,0,213,0,15
1590 DATA 12,1,225,0,5
1600 DATA 253,0,253,0,10
1610 DATA 225,0,82,1,17
1620 DATA 225,0,169,0,10
1630 DATA 12,1,225,0,10
1640 DATA 82,1,82,1,20
1650 DATA 82,1,123,1,10
1660 DATA 142,0,170,1,30
1670 DATA 142,0,123,1,5
1680 DATA 142,0,82,1,10
1690 DATA 142,0,82,1,15
1700 DATA 159,0,123,1,5
1710 DATA 169,0,170,1,10
1720 DATA 189,0,28,1,30
1730 DATA 189,0,253,0,5
1740 DATA 225,0,225,0,10
1750 DATA 142,0,225,0,15
1760 DATA 253,0,253,0,5
1770 DATA 225,0,28,1,10
1780 DATA 213,0,253,0,30
1790 DATA 225,0,225,0,5
1800 DATA 253,0,213,0,5
1810 DATA 253,0,189,0,5
1820 DATA 12,1,169,0,15
1830 DATA 45,1,169,0,5
1840 DATA 12,1,82,1,10
1850 DATA 253,0,253,0,30
1860 DATA 253,0,126,0,10
1870 DATA 253,0,169,0,10
1880 DATA 253,0,253,0,30
1890 DATA 8,0,9,0,10,0
1900 END
```

A linha 10 protege a memória acima de – 14501 para ali colocar a tabela. O laço **FOR...NEXT** entre as linhas 20 e 40 usa **READ** para obter os números da tabela e **POKE** para transferi-los para a memória.

Cada linha **DATA** contém os dados necessários para um acorde. Os números são respectivamente: byte menos significativo da tonalidade do canal A, byte mais significativo dessa mesma tonalidade, byte menos significativo da tonalidade do canal B, byte mais significativo dessa mesma tonalidade e duração do acorde.

Não se preocupe com os valores que determinam as tonalidades das notas. Em um próximo artigo, apresentaremos uma tabela para a conversão dos parâmetros da instrução **PLAY** nos valores de registros do PSG.



O JOGO DO OTELO (2)

Apresentamos neste artigo a segunda metade do jogo do Otelo. Depois de digitá-la, completando o programa, desenvolva algumas jogadas inteligentes... e derrote o computador!

Aqui está a última parte do jogo do Otelo. Após digitar as linhas necessárias para completar a movimentação do jogador, a rotina de movimentação do computador e o final da rotina de jogo, você poderá se considerar pronto para testar suas habilidades, desafiando o seu computador.

VERIFICAÇÃO DAS POSIÇÕES



```
2090 IF NF=1 THEN 2120
2100 COLOR 1:LINE(0,182)-(255,1
91),PSET,BF:DRAW"C4S8BM0,182":A
$="LONGE DA JOGADA":GOSUB 9300:
FOR F=1 TO 900:NEXT F
2110 GOTO 2000
2120 RF=0:FOR Q=1 TO 8:IF C(Q)=
0 THEN 2170
2130 XP=X:YP=Y
2140 XP=XP+D(Q,1):YP=YP+D(Q,2):
IF XP=0 OR XP=9 OR YP=0 OR YP=9
 THEN C(Q)=0:GOTO 2170
2145 IF B(XP,YP)=2 THEN 2140
2150 IF B(XP,YP)=1 THEN RF=1:GO
TO 2170
2160 IF B(XP,YP)=0 THEN C(Q)=0
2170 NEXT Q
2180 IF RF=1 THEN 2210
2190 COLOR1:LINE(0,182)-(255,19
1),PSET,BF:DRAW"S8C4BM0,182":AS
="JOGADA NAO GANHA A TRILHA":GO
SUB 9300:FOR F=1 TO 900:NEXTF
2200 GOTO 2000
2210 FOR Q=1 TO 8:IF C(Q)=0 THE
N 2250
2220 XP=X+D(Q,1):YP=Y+D(Q,2)
2230 IF B(XP,YP)=1 THEN 2250
2240 B(XP,YP)=1:XP=XP+D(Q,1):YP
=YP+D(Q,2):GOTO 2230
2250 NEXT Q
2260 B(X,Y)=1
2270 CP=2:RETURN
```



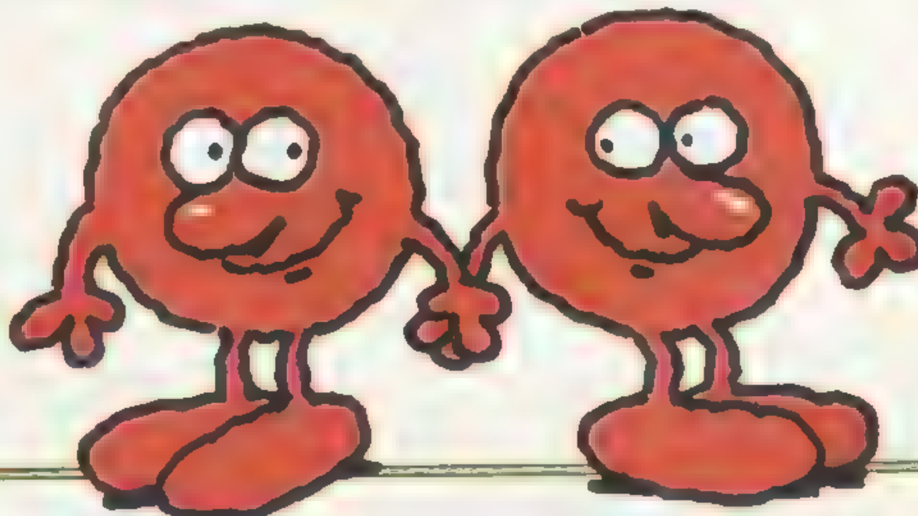
```
2090 IF NF=1 THEN  GOTO 2120
2100 PRINT AT 17,0;"LONGE DA JO
GADA": FOR F=1 TO 500: NEXT F
2110 PRINT AT 17,0;"
      ": GOTO 2000
2120 LET RF=0: FOR Q=1 TO 8: IF
 C(Q)=0 THEN  GOTO 2170
2130 LET XP=X: LET YP=Y
2140 LET XP=XP+D(Q,1): LET YP=Y
P+D(Q,2): IF XP=0 OR XP=9 OR YP
=0 OR YP=9 THEN  LET C(Q)=0: GO
TO 2170
2145 IF B(XP,YP)=2 THEN  GOTO 2
140
2150 IF B(XP,YP)=1 THEN  LET RF
=1: GOTO 2170
2160 IF B(XP,YP)=0 THEN  LET C(
Q)=0
2170 NEXT Q
2180 IF RF=1 THEN  GOTO 2210
2190 PRINT AT 17,0,"JOGADA NAO
GANHA A TRILHA": FOR F=1 TO 500
: NEXT F
2200 PRINT AT 17,0;"
                  ": GOTO 2000
2210 FOR Q=1 TO 8: IF C(Q)=0 TH
EN  GOTO 2250
2220 LET XP=X+D(Q,1): LET YP=Y+
D(Q,2)
2230 IF B(XP,YP)=1 THEN  GOTO 2
250
2240 LET B(XP,YP)=1: LET XP=XP+
D(Q,1): LET YP=YP+D(Q,2): GOTO
2230
2250 NEXT Q
2260 LET B(X,Y)=1
2270 LET CP=2: RETURN
```



```
2090 IFNF=1 THEN 2120
2100 LINE(0,182)-(255,191),1,BF
:DRAW"C15S8BM50,182":A$="LONGE
DA JOGADA":GOSUB9300:FORF=1TO90
0:NEXTF
2110 GOTO 2000
2120 RF=0:FORQ=1 TO 8:IFC(Q)=0
THEN2170
2130 XP=X:YP=Y
2140 XP=XP+D(Q,1):YP=YP+D(Q,2):
IFXP=0 OR XP=9 OR YP=0 OR YP=9
THENC(Q)=0:GOTO2170
2145 IF B(XP,YP)=2 THEN 2140
2150 IF B(XP,YP)=1 THEN RF=1:GO
TO 2170
2160 IF B(XP,YP)=0 THEN C(Q)=0
2170 NEXTQ
2180 IF RF=1 THEN 2210
2190 LINE(0,182)-(255,191),1,BF
:DRAW"C15S8BM10,182":A$="JOGADA
 NAO GANHA A TRILHA":GOSUB9300:
FORF=1TO900:NEXTF
2200 GOTO 2000
2210 FORQ=1TO8:IFC(Q)=0THEN2250
2220 XP=X+D(Q,1):YP=Y+D(Q,2)
2230 IF B(XP,YP)=1 THEN2250
2240 B(XP,YP)=1:XP=XP+D(Q,1):YP
=YP+D(Q,2):GOTO2230
2250 NEXTQ
2260 B(X,Y)=1
2270 CP=2:RETURN
```



```
2090  IF NF = 1 THEN 2120
2100  VTAB (17): PRINT "JOGADA
LONGE DAS MINHAS PECAS": FOR F
= 0 TO 1500: NEXT
2110  VTAB (17): PRINT "
                        ": GOTO 2
000
2120 RF = 0: FOR Q = 1 TO 8: IF
 C(Q) = 0 THEN 2170
2130 XP = X:YP = Y
2140 XP = XP + D(Q,1):YP = YP +
 D(Q,2): IF (XP = 0 OR XP = 9)
OR (YP = 0 OR YP = 9) THEN C(Q)
 = 0: GOTO 2170
2145  IF B(XP,YP) = 2 THEN 2140
2150  IF B(XP,YP) = 1 THEN RF =
 1: GOTO 2170
2160  IF B(XP,YP) = 0 THEN C(Q)
 = 0
2170  NEXT
2180  IF RF = 1 THEN 2210
```



- MOVIMENTAÇÃO DO JOGADOR: VERIFICAÇÃO DAS POSIÇÕES
- ROTINA DE MOVIMENTAÇÃO DO COMPUTADOR
- FINAL DA ROTINA DE JOGO
- ANÚNCIO DO VENCEDOR
- OPÇÃO PARA NOVA PARTIDA

```
2190  VTAB (17)· PRINT "SUA JOG
ADA NAO GANHA A TRILHA": FOR F
= 0 TO 1500· NEXT
2200  VTAB (17)· PRINT "
                       ": GOTO 2
000
2210  FOR Q = 1 TO 8· IF C(Q) =
 0 THEN 2250
2220 XP = X + D(Q,1):YP = Y + D
(Q,2)
2230  IF B(XP,YP) = 1 THEN 2250
2240 B(XP,YP) = 1:XP = XP + D(Q
,1).YP = YP + D(Q,2)  GOTO 2230
2250  NEXT Q
2260 B(X,Y) = 1
2270 CP = 2· RETURN
```

Antes de saltar para a linha 2120, o indicador NF vê se há alguma peça do computador em um quadrado adjacente (linha 2090). Caso exista, a linha 2100 imprime uma mensagem de erro. As linhas 2120 a 2170 verificam se a jogada encosta a peça em alguma fileira de peças do adversário.

A linha 2140 verifica se a posição checada no passo anterior está dentro dos limites do tabuleiro. Se não estiver, esta posição será abandonada e a próxima, testada. Na linha 2145, testamos se a peça em exame pertence, de fato, ao computador. Em caso afirmativo, o programa volta para a linha 2140 e atualiza a posição.

A linha 2180 examina o resultado da última operação. Se a fileira foi encontrada, o programa salta para a linha 2210. As linhas 2190 e 2200 imprimem uma mensagem no caso de uma fileira não ser ladeada. O programa volta para a linha 2000.

A rotina que descreve a movimentação do jogador está contida entre as linhas 2210 e 2260 do programa. O teste **C(Q) = 0**, da linha 2210, verifica se uma fileira do adversário foi encontrada, por meio de um ciclo que vai de 1 a 8. Se não existir uma fileira naquela direção **(C(Q) = 0)**, o programa salta para a linha 2250 e verifica o **Q** seguinte. A posição do primeiro quadrado da fileira a ser tomada — linha 2040 — é atribuída a **XP** e **YP**.

Na linha 2230, o computador testa se o quadrado em questão é o último da fileira, encerrando-a, portanto. Em caso afirmativo, o programa salta para a linha 2250. Na linha 2240, o programa atribui "um" ao quadrado e, em seguida, volta para a linha 2230, onde testa o próximo quadrado.

Na linha 2260, o quadrado do jogador recebe o valor inicial "um". O indicador **CP** recebe o valor "dois" para o computador e o programa retorna (por meio da linha 2270).

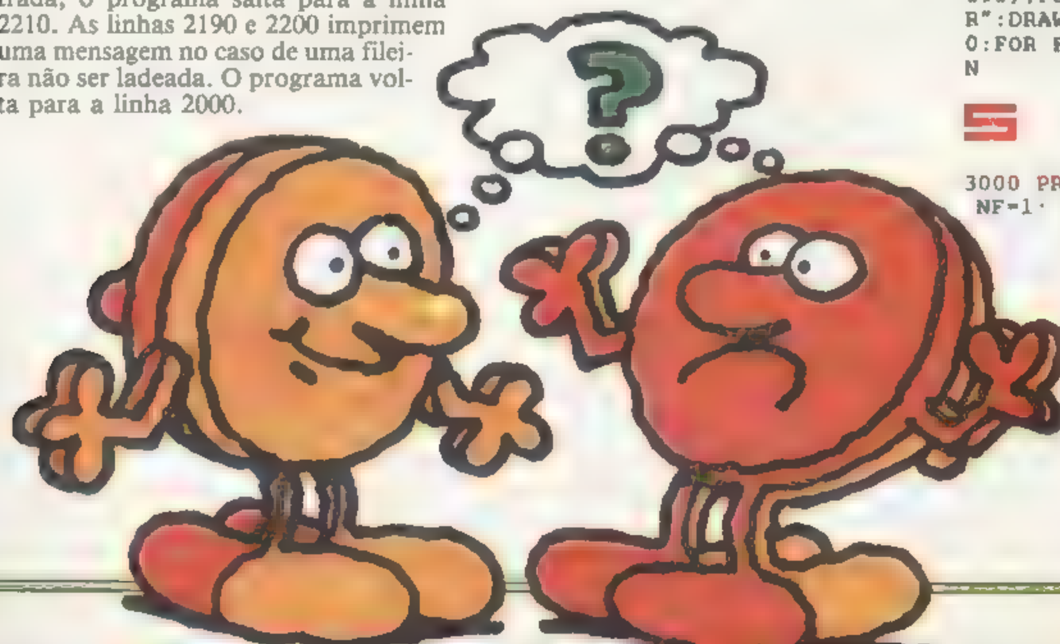
A JOGADA DO COMPUTADOR



```
3000 COLOR 1:LINE(0,182)-(255,1
91),PSET,BF:A$="MOVIMENTO DO CO
MPUTADOR":DRAW"S8C3BM26,182":GO
SUB 9300:NF=1:MX=0:FOR X=1 TO 8
:FOR Y=1 TO 8
3010 IF B(X,Y)<>0 THEN 3070
3020 FOR F=1 TO 8:XP=X:YP=Y:DX=
D(F,1):DY=D(F,2):RF=0
3030 XP=XP+DY:YP=YP+DX:IF XP=0
OR XP=9 OR YP=9 THEN 3060
3040 IF B(XP,YP)=1 THEN RF=1:GO
TO 3030
3050 IF B(XP,YP)=2 AND RF=1 THE
N N(NF)=F:X(NF)=X:Y(NF)=Y:NF=NF
+1:F=9
3060 NEXT F
3070 NEXT Y,X:NF=NF-1
3075 IF NF=0 THEN 3300
3080 FOR F=1 TO NF:X=X(F):Y=Y(F
):DX=D(N(F),1):DY=D(N(F),2):CF=
0
3090 X=X+DY:Y=Y+DX:IF B(X,Y)=1
THEN CF=CF+1:GOTO 3090
3100 IF CF>MX THEN MX=CF:MF=F
3110 NEXT F
3180 FOR F=1 TO 8:X=X(MF):Y=Y(M
F):DX=D(F,1):DY=D(F,2)
3190 X=X+DY:Y=Y+DX
3195 IF X<1 OR X>8 OR Y<1 OR Y>
8 THEN 3260
3200 IF B(X,Y)=1 THEN 3190
3210 IF B(X,Y)=2 THEN 3230
3220 IF B(X,Y)=0 THEN 3260
3230 X=X(MF):Y=Y(MF)
3235 B(X,Y)=2:X=X+DY:Y=Y+DX
3240 IF B(X,Y)=2 THEN 3260
3250 GOTO 3235
3260 NEXT F
3265 DRAW"S16:C2;BM118,150"+NU$
(X(MF))+NU$(Y(MF))+"S8"
3270 CP=1:RETURN
3300 COLOR 1:LINE (0,182)-(255,
191),PSET.BF·A$="NAO POSSO MOVE
R":DRAW"S8C4BM50,182":GOSUB 930
0:FOR F=1 TO 50:NEXT:CP=1:RETUR
N
```



```
3000 PRINT ''"PENSANDO...": LET
 NF=1· LET MX=0: FOR X=1 TO 8:
```



```
FOR Y=1 TO 8
3010 IF B(X,Y)<>0 THEN  GOTO 30
70
3020 FOR F=1 TO 8: LET XP=X: LE
T YP=Y: LET DX=D(F,1): LET DY=D
(F,2): LET RF=0
3030 LET XP=XP+DY: LET YP=YP+DX
: IF XP=0 OR XP=9 OR YP=0 OR YP
=9 THEN  GOTO 3060
3040 IF B(XP,YP)=1 THEN  LET RF
=1: GOTO 3030
3050 IF B(XP,YP)=2 AND RF=1 THE
N  LET N(NF)=F: LET X(NF)=X: LE
T Y(NF)=Y: LET NF=NF+1: LET F=9
3060 NEXT F
3070 NEXT Y: NEXT X: LET NF=NF-
1
3075 IF NF=0 THEN  GOTO 3300
3080 FOR F=1 TO NF: LET X=X(F):
 LET Y=Y(F): LET DX=D(N(F),1):
LET DY=D(N(F),2): LET CF=0
3090 LET X=X+DY: LET Y=Y+DX: IF
 B(X,Y)=1 THEN  LET CF=CF+1: GO
TO 3090
```

MICRO DICAS

PROGRAMAÇÃO DE MATRIZES

A programação de jogos de tabuleiro em BASIC utiliza quase sempre a mesma técnica: a da programação de matrizes. Uma matriz retangular, dividida em linhas e colunas, é a estrutura de dados ideal para a programação e representação de tabuleiros de xadrez, damas, Otelo, Reversi, Trilha, jogo da velha e outros. Entretanto, jogos que não seguem o esquema de "quadradinhos" identificados em linhas e colunas — como gamão, gomoku e ludo também podem ser programados através de matrizes.

Em um jogo de tabuleiro retangular, geralmente uma só matriz bidimensional não basta para armazenar todos os dados exigidos. Tomemos como exemplo o xadrez. A cor dos quadrados do tabuleiro pode ser computada por meio de uma equação simples. É mais fácil, porém, armazenar esse tipo de dado com códigos numéricos, em uma matriz COR (8,8). Precisaremos, ainda, de uma matriz para indicar as peças que ocupam cada quadrado (novamente, podemos utilizar códigos, como pião = 1, torre = 2 e assim por diante). Outras matrizes serão necessárias para dados referentes a movimentos, capturas etc.

Tudo isso torna a programação de jogos de tabuleiro mais complexa do que parece. Portanto, se você pretende testar sua habilidade como programador, escolha um jogo bem fácil para começar!

```
3100 IF CF>MX THEN  LET MX=CF:
LET MF=F
3110 NEXT F
3180 FOR F=1 TO 8: LET X=X(MF):
 LET Y=Y(MF): LET DX=D(F,1): LE
T DY=D(F,2)
3190 LET X=X+DY: LET Y=Y+DX
3195 IF X<1 OR X>8 OR Y<1 OR Y>
8 THEN  GOTO 3260
3200 IF B(X,Y)=1 THEN  GOTO 319
0
3210 IF B(X,Y)=2 THEN  GOTO 323
0
3220 IF B(X,Y)=0 THEN  GOTO 326
0
3230 LET X=X(MF): LET Y=Y(MF)
3235 LET B(X,Y)=2: LET X=X+DY:
LET Y=Y+DX
3240 IF B(X,Y)=2 THEN  GOTO 326
0
3250 GOTO 3235
3260 NEXT F
3265 PRINT X(MF),Y(MF): INPUT A
$
3270 LET CP=1: RETURN
3300 PRINT AT 17,0;"NAO POSSO J
OGAR": FOR F=1 TO 500: NEXT F
3305 PRINT AT 17,0;"
                              "
3310 LET CP=1
3320 RETURN
```



```
3000 LINE(0,182)-(255,191),1,BF
:DRAW"S8C15BM30,182":A$="COMPUT
ADOR JOGANDO":GOSUB9300:NF=1:MX
=0:FORX=1TO8:FORY=1TO8
3010 IFB(X,Y)<>0THEN3070
3020 FORF=1TO8:XP=X:YP=Y:DX=D(F
,1):DY=D(F,2):RF=0
3030 XP=XP+DY:YP=YP+DX:IFXP=0 O
R XP=9 OR YP=0 OR YP=9 THEN3060
3040 IFB(XP,YP)=1THENRF=1:GOTO3
030
3050 IFB(XP,YP)=2 AND RF=1 THEN
 N(NF)=F:X(NF)=X:Y(NF)=Y:NF=NF+
1:F=9
3060 NEXTF
3070 NEXTY,X:NF=NF-1
3075 IF NF=0 THEN 3300
3080 FORF=1TONF:X=X(F):Y=Y(F):D
X=D(N(F),1):DY=D(N(F),2):CF=0
3090 X=X+DY:Y=Y+DX:IFB(X,Y)=1TH
ENCF=CF+1:GOTO3090
3100 IFCF>MX THEN MX=CF:MF=F
3110 NEXTF
3180 FORF=1TO8:X=X(MF):Y=Y(MF):
DX=D(F,1):DY=D(F,2)
3190 X=X+DY:Y=Y+DX
3195 IFX<1 OR X>8 OR Y<1 OR Y>8
 THEN3260
3200 IF B(X,Y)=1 THEN 3190
3210 IF B(X,Y)=2 THEN 3230
3220 IF B(X,Y)=0 THEN 3260
3230 X=X(MF):Y=Y(MF)
3235 B(X,Y)=2:X=X+DY:Y=Y+DX
3240 IF B(X,Y)=2 THEN 3260
3250 GOTO 3235
3260 NEXTF
3265 DRAW"S16BM114,154"+NU$(X(M
```



```
F))+NU$(Y(MF))+"S8"
3270 CP=1:RETURN
3300 LINE(0,182)-(255,191),1,BF
:DRAW"C15S8BM50,182":A$="NAO PO
SSO JOGAR":GOSUB9300:FORF=1TO90
0:NEXTF:CP=1:RETURN
```



```
3000 NF = 1:MX = 0: FOR X = 1 T
O 8: FOR Y = 1 TO 8
3010  IF B(X,Y) <  > 0 THEN 307
0
3020  FOR F = 1 TO 8:XP = X:YP
= Y:DX = D(F,1):DY = D(F,2):RF
= 0
3030 XP = XP + DY:YP = YP + DX:
 IF (XP = 0 OR XP = 9) OR (YP =
 0 OR YP = 9) THEN 3060
3040  IF B(XP,YP) = 1 THEN RF =
 1: GOTO 3030
3050  IF B(XP,YP) = 2 AND RF =
1 THEN N(NF) = F:X(NF) = X:Y(NF
) = Y:NF = NF + 1:F = 9
3060  NEXT F
3070  NEXT Y: NEXT X:NF = NF -
1
3075  IF NF = 0 THEN 3300
```



```
3080  FOR F = 1 TO NF:X = X(F):
Y = Y(F):DX = D(N(F),1):DY = D(
N(F),2):CF = 0
3090 X = X + DY:Y = Y + DX: IF
B(X,Y) = 1 THEN CF = CF + 1: GO
TO 3090
3100  IF CF > MX THEN MX = CF:M
F = F
3110  NEXT
3180  FOR F = 1 TO 8:X = X(MF):
Y = Y(MF):DX = D(F,1):DY = D(F,
2)
3190 X = X + DY:Y = Y + DX
3195  IF X < 1 OR X > 8 OR Y <
1 OR Y > 8 THEN 3260
3200  IF B(X,Y) = 1 THEN 3190
3210  IF B(X,Y) = 2 THEN 3230
3220  IF B(X,Y) = 0 THEN 3260
3230 X = X(MF):Y = Y(MF)
3235 B(X,Y) = 2:X = X + DY:Y =
Y + DX
3240  IF B(X,Y) = 2 THEN 3260
3250  GOTO 3235
3260  NEXT
3265  VTAB (17): PRINT "MINHA J
OGADA (LINHA,COLUNA) = ";X(MF);
",";Y(MF): FOR F = 0 TO 3000: N
EXT
3270  VTAB (17): PRINT "

":CP = 1: RETURN
3300  VTAB (17): PRINT "NAO POS
SO JOGAR": FOR F = 0 TO 1500: N
EXT
3310  VTAB (17): PRINT "
      ":CP = 1: RETURN
```

No começo, o número de quadrados em uma fileira é um, e o número máximo de peças, zero (linha 3000). Dois laços — com variáveis de controle **X** e **Y** — são iniciados, com a função de procurar espaços vazios no tabuleiro. Esta é a parte do programa que consome maior tempo. No decorrer do jogo, à medida que o número de quadrados vazios diminui, o tempo de jogada do computador também vai diminuindo.

A linha 3010 verifica se o quadrado está vazio. Se não estiver, o computador salta os próximos comandos (linha 3070). As linhas que vão de 3020 a 3060 verificam se o quadrado está no fim de uma fileira que poderá ser tomada pelo computador. **XP** e **YP** são usados da mesma maneira que anteriormente. **DX** e **DY** representam os conteúdos do vetor direção **D()** e economizam bastante espaço de memória.

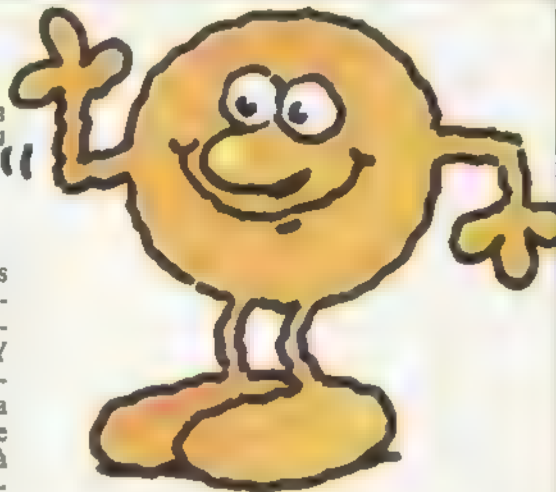
A linha 3030 checa se o quadrado que vai ser testado está dentro dos limites do tabuleiro. Se ele não estiver, a próxima direção será testada. Se o quadrado em exame pertencer ao jogador, a rotina voltará para a linha 3030 para verificar se o quadrado seguinte está ocupado por ele.

A linha 3050 examina o tabuleiro, para saber se está sendo ocupado pelo computador. Caso esteja, e se alguma fileira tiver sido encontrada, as posições iniciais serão gravadas em **X()** e **Y()**, e o número da direção será gravado em **N()**. O contador que indica o número de coordenadas encontradas é também incrementado.

Apenas a primeira fileira será "memorizada", para garantir que a busca leve o menor tempo possível.

As linhas 3080 a 3110 encontram a jogada capaz de fornecer o placar mais longo em uma linha reta. Um ciclo que varia de 1 até **NF** (número de quadrados encontrados) é acionado. **X** e **Y** são equacionados com **X(F)** e **Y(F)**. As coordenadas de direção (**DX** e **DY**) recebem valores iniciais correspondentes às direções indicadas por **N(F)**. Um contador temporário (**CF**) é zerado a cada execução do ciclo.

A linha 3090 verifica se a peça que foi testada é a do jogador. Em caso afirmativo, **CF** é incrementado e o próximo quadrado naquela direção torna-se



objeto de teste. A linha 3100 compara o número encontrado (**CF**) com o valor máximo anterior (**MX**). Se o primeiro for maior, **MX** assume o valor de **CF** e **MF**, as coordenadas da melhor peça encontrada, indicada pelo valor do contador do laço, ou seja, **F**.

As linhas 3180 a 3260 executam a rotina de movimentação. A linha 3180 inicia um ciclo que vai de 1 a 8, de forma que todas as fileiras possam ser encontradas.

As variáveis **X**, **Y**, **DX** e **DY** recebem valores iniciais conforme foi explicado anteriormente. A linha 3195 checa se **X** e **Y** ainda estão no tabuleiro. Caso eles não estejam, o programa salta imediatamente para a linha 3260, que contém uma instrução **NEXT**.

A linha 3200 verifica se o jogador está ocupando um quadrado. Se estiver, a rotina salta para tentar o próximo quadrado da fileira. Nenhum quadrado é alterado, pois a rotina está apenas realizando testes.

Se a fileira terminar em um quadrado ocupado pelo computador, **X** e **Y** são reinicializadas e as linhas 3235 a 3250 alteram todos os quadrados na fileira. Se um quadrado vazio for encontrado, tenta-se uma nova direção.

Assim que a rotina de movimentação tiver decidido para qual quadrado irá se dirigir, a linha 3265 imprimirá as coordenadas e aguardará que a tecla <ENTER> seja pressionada.

Na vez do jogador, o indicador **CP** começa valendo "um" (linha 3270) e, depois, volta para o ciclo principal.

FIM DE JOGO



```
4000 IF PS>CS THEN 5000
4010 IF PS=CS THEN 6000
```



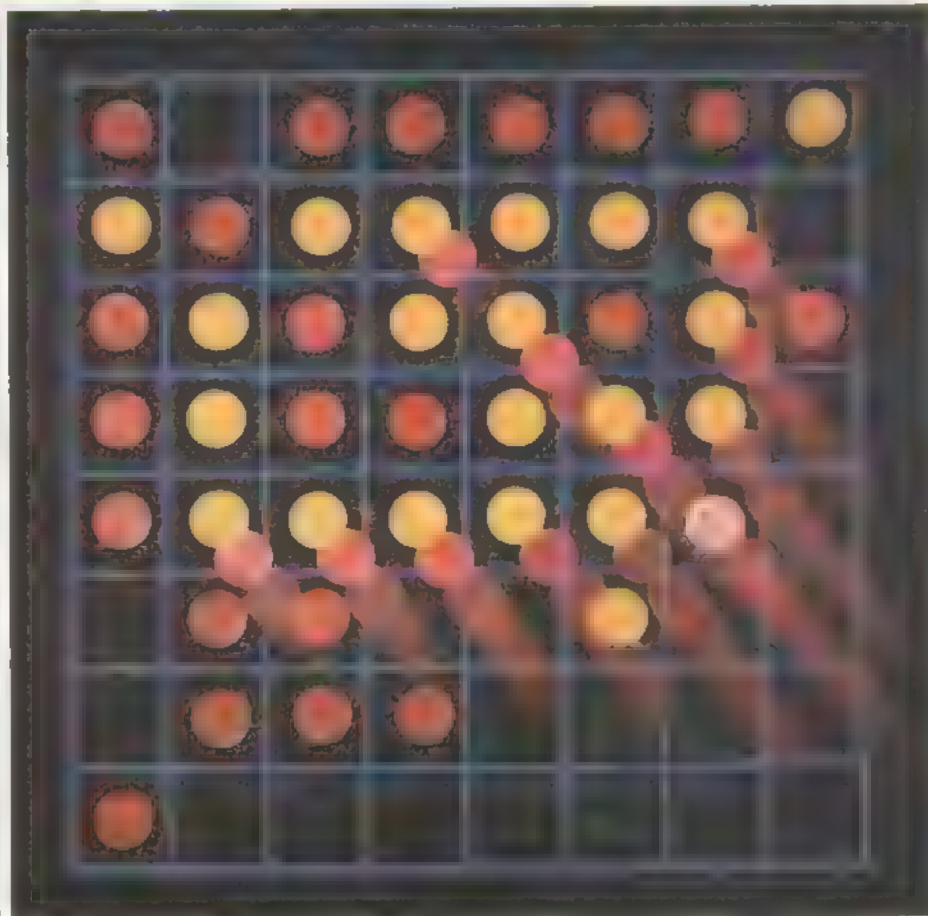
Como adaptar um jogo que utiliza peças coloridas para um micro com vídeo monocromático?

Dependendo do tipo de jogo, a adaptação não é nada fácil. Mas existem algumas alternativas cujo resultado é bastante razoável:

- Para diferenciar as peças dos oponentes, utilize recursos gráficos como hachurados, quadriculados ou pontilhados, cujo efeito é bem visível em preto e branco. Você só terá problemas se as peças forem pequenas a ponto de não haver espaço para nenhum tipo de desenho que as identifique.

- Também como elemento de diferenciação, faça uso do vídeo inverso/vídeo direto — por exemplo, pedras pretas em um quadrado preto, pedras brancas em um quadrado branco e pedras sobre quadrados de outra cor.

- Se o seu computador possui um vídeo monocromático multitonal (isto é, em que as cores têm correspondência em diversas tonalidades de verde ou cinza), identifique as peças dos oponentes utilizando intensidades ou brilhos diferentes.



```
4015 A$="FOI FACIL"
4020 COLOR 1:LINE(0,182)-(255,1
91),PSET,BF:DRAW"S8C2BM70,182":
GOSUB 9300
4025 FOR F=1 TO 1500:NEXT F
4030 COLOR 1:LINE(0,182)-(255,1
91),PSET,BF:A$="QUER JOGAR OUTR
A VEZ ?":DRAW"C3BM0,182":GOSUB
9300
4040 A$=INKEY$:IF A$<>"S" AND A
$<>"N" THEN 4040
4050 IF A$="S" THEN RUN
4060 END
5000 A$="VOCE TEVE SORTE"
5010 GOTO 4020
6000 A$="EMPATAMOS":GOTO 4020
```



```
4000 IF PS>CS THEN  GOTO 5000
4010 IF PS=CS THEN  GOTO 6000
4020 PRINT AT 17,0; INK 2;"FOI
FACIL !"
4030 PRINT "QUER JOGAR OUTRA VE
Z ? (S/N)"
4040 LET A$=INKEY$: IF A$<>"S"
AND A$<>"N" THEN  GOTO 4040
4050 IF A$="S" THEN  RUN
4060 STOP
5000 PRINT AT 17,0; INK 2;"VOCE
 TEVE SORTE !"
5010  GOTO 4030
6000 PRINT AT 17,0; INK 2;"NOS
EMPATAMOS ...                PRE
CISO PRATICAR MAIS !": GOTO 403
0
```



```
4000 IF PS>CS THEN 5000
4010 IF PS=CS THEN 6000
4015 A$="FOI MUITO FACIL"
4020 LINE(0,182)-(255,191),1,BF
:DRAW"C15S8BM70,182":GOSUB9300
4025 FORF=1TO1500:NEXTF
4030 LINE(0,182)-(255,191),1,BF
:A$="QUER JOGAR NOVAMENTE":DRAW
"C15BM30,182":GOSUB9300
4040 A$=INKEY$:IFA$<>"S" AND A$
<>"s" AND A$<>"N" AND A$<>"n" T
HEN 4040
4050 IF A$="S" OR A$="s" THEN R
UN
4060 END
5000 A$="VOCE TEVE SORTE"
5010 GOTO 4020
6000 A$="UM BELO EMPATE":GOTO40
20
```



```
4000  IF PS > CS THEN 5000
4010  IF PS = CS THEN 6000
4020  VTAB (17): PRINT "FOI FAC
IL !"
4030  INPUT "QUER JOGAR NOVAMEN
TE (S/N) ";A$
4040 A$ =  LEFT$ (A$,1): IF A$
< > "S" THEN  END
4050  RUN
4060  END
5000  VTAB (17): PRINT "VOCE TE
VE SORTE DESTA VEZ !"
5010  GOTO 4030
6000  VTAB (17): PRINT "EMPATAM
OS, PRECISO TREINAR MAIS !": GO
TO 4030
```

A rotina de fim de jogo começa na linha 4000, que verifica se o jogador venceu, comparando PS com CS. O programa salta para a linha 5000 para imprimir a mensagem de vitória. A linha 4010 detecta a ocorrência de empate, e a mensagem correspondente é impressa pela linha 6000. Se o computador tiver vencido, o programa atinge a linha 4020 e exibe uma mensagem para o jogador. As linhas restantes oferecem a opção para uma outra partida.

FIGURAS GEOMÉTRICAS

- AS SEÇÕES DO CONE
- DESENHE CÍRCULOS, ELIPSES, PARÁBOLAS E HIPÉRBOLES
- ROTAÇÃO DE CURVAS
- APLICAÇÕES PRÁTICAS

Forma geométrica das mais fascinantes, o cone é responsável pela geração de toda uma família de curvas. Eis aqui alguns programas para explorar as propriedades dessa figura.

Criadores da geometria, os gregos antigos consideravam as curvas — pelas quais eram fascinados — tanto mais importantes quanto mais simples e elegantes eles fossem. Assim, quando se descobriu que toda uma família de curvas — conhecidas como seções cônicas — poderia ser obtida simplesmente cortando-se um cone de diferentes maneiras, pareceu óbvio que esses sólidos geométricos deviam ter uma significação especial. Realmente, dependendo de como se secciona o cone, pode-se obter um círculo, uma elipse, uma parábola ou uma hipérbole.

Na verdade, o traço marcante dessas curvas está no fato de que elas não são meras abstrações matemáticas, mas aparecem a todo momento no nosso cotidiano e proporcionam uma descrição exata de fenômenos físicos.

Existem, certamente, outras curvas simples encontradas na natureza que não são seções de um cone. A forma produzida por uma corda ou corrente presa por dois pontos nas extremidades é um exemplo. Conhecida como catenária, essa curva difere ligeiramente da parábola. Entretanto, as equações que servem para descrever as duas formas são completamente distintas. As seções cônicas aparecem freqüentemente relacionadas ao modo como as coisas se movem (um objeto lançado sobre a superfície da Terra, por exemplo, executa uma curva parabólica). Assim, elas são necessárias para uma simulação convincente e exata desses movimentos.

Algumas curvas também são úteis para definir objetos tridimensionais. Os cortes de um cone têm obviamente duas dimensões, mas podem ser rodados em seus eixos para produzir uma forma de três dimensões. Assim, o círculo se transforma em uma esfera, com um sem número de aplicações; a parábola, em um parabolóide, usado em objetos como faróis de automóveis, espelhos de telescópios, fornos solares etc.

A primeira parte deste artigo descreve cada curva e como desenhá-la na tela do computador, enquanto a segunda mostra como usá-la em simulações como a trajetória de um balde (ou outro objeto) pendurado em uma escada que desliza e acaba caindo (uma elipse) ou a curva descrita por uma pessoa cruzando um rio a nado (uma parábola).

Veremos também como desenhar belas formas usando a hipérbole e a elipse.



COMO CORTAR O CONE

As quatro curvas obtidas quando se secciona um cone — o círculo, a elipse, a parabola e a hiperbole — diferem totalmente entre si. Foram estudadas em detalhe pela primeira vez pelo grego Apolônio, por volta de 200 a.C.

O ponto inicial é: se um par de retas que se cruzam — como um X — roda em torno de um eixo de simetria, gera-se um duplo cone (veja os desenhos destas páginas) que pode ser cortado por um plano de quatro maneiras.

Se um corte é feito em ângulo reto ao eixo de simetria, a seção resultante é um círculo.

Um corte feito em um ângulo entre 90° e metade do ângulo formado pelas linhas que geraram o cone (conhecido como ângulo semivertical do cone) produz uma elipse.

Um plano que faz com o eixo um ângulo igual ao ângulo semivertical nos dá uma parabola. Já um plano que faz com o eixo um ângulo menor que o semivertical cria uma seção em duas partes chamada hipérbole. Ela tem duas partes porque nosso plano corta tanto o cone de cima como o de baixo (lembre-se de que o duplo cone foi gerado pela rotação de um X).

Existem dois casos especiais. Se, por exemplo, o corte é feito por um plano que passa pelo eixo, isto é, se os cones são divididos verticalmente pela metade, serão obtidas duas linhas retas — aquelas que foram usadas para gerar a figura. Elas configuram na realidade um caso especial de hipérbole. Por outro lado, se um plano cortar os cones pelo vértice, fazendo um ângulo reto com o eixo vertical, tudo o que se tem é um ponto, que é um círculo de raio zero.

Os desenhos destas páginas devem deixar claro como tais formas são obtidas. Se quiser, você também pode cortar cones feitos de material de fácil manipulação — como isopor ou papel — e verificar o que obtém. Um cone duplo só será necessário se você quiser ter uma hipérbole de verdade, visto que ela sempre tem duas partes.

CURVAS E MAIS CURVAS

As curvas são geradas por equações simples, algumas das quais já foram abordadas em outros programas. Elas serão sempre desenhadas na tela de alta resolução do seu micro. Vejamos agora algumas delas. Comecemos pelo círculo, uma das curvas mais simples.

O CÍRCULO

A equação de um círculo é dada por:

X = A*COS teta
Y = A*SEN teta

onde A é o raio, X e Y, um ponto qualquer da circunferência, e teta, o ângulo formado com uma reta prefixada, em geral o eixo X.

O primeiro programa desenha um círculo de raio A no meio da tela:

S

```
10 CLS
15 LET a=70
25 LET x=a: LET y=0
30 PLOT 127+x,70+y
40 FOR t=0 TO 2*PI STEP .2
50 LET x=a*COS t: LET y=a*SIN
 t
60 DRAW x-PEEK 23677+127,y-
PEEK 23678+70
70 NEXT t
```

T

```
10 PMODE 4:PCLS:SCREEN 1,1
15 A=60
20 C=ATN(1)/45
30 LINE-(127+A,95),PRESET
40 FOR TH=0 TO 360 STEP 10
50 X=A*COS(TH*C):Y=A*SIN(TH*C)
60 LINE -(127+X,95+Y),PSET
70 NEXT TH
80 GOTO 80
```

M

```
10 COLOR 15,4,4:SCREEN2
15 A=60
20 C=ATN(1)/45
30 LINE -(127+A,95),4
40 FOR TH=0 TO 360 STEP 10
50 X=A*COS(TH*C):Y=A*SIN(TH*C)
60 LINE -(127+X,95+Y),15
70 NEXT
80 GOTO 80
```

```
10  HOME : HGR : HCOLOR= 3
15 A = 60
20 C =  ATN (1) / 45
30  HPLOT 127 + A,95
40  FOR TH = 0 TO 360 STEP 10
50 X = A *  COS (TH * C):Y = A
*  SIN (TH * C)
60  HPLOT  TO 127 + X,95 + Y
70  NEXT
```

O laço **FOR...NEXT** das linhas 40 a 70 é a parte do programa que desenha o círculo, traçando repetidamente segmentos de reta a intervalos de 10° (ou 0.2 radianos). O raio do círculo é definido na linha 15.

A ELIPSE

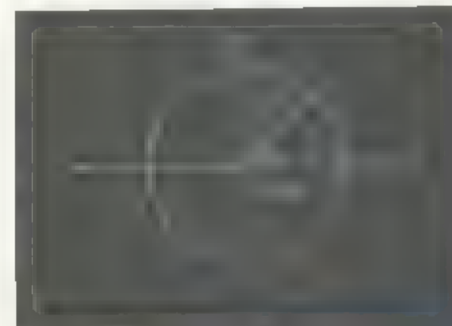
A equação da elipse é muito parecida com a do círculo. Para uma elipse com o eixo maior 2A e o menor 2B, a posição do ponto da periferia é:

X = A*COS teta
Y = B*SEN teta

A forma da elipse — ou seja, suas proporções de características mais ou menos achatadas — é determinada por A e B. Altere estas linhas:

S

```
16 LET b=40
50 LET x=a*COS t: LET y=b*SIN
t
```



O círculo





```
16 B=30
50 X=A*COS(TH*C):Y=B*SIN(TH*C)
```



```
16 B=30
50 X=A*COS(TH*C):Y=B*SIN(TH*C)
```



```
16 B = 30
50 X = A * COS (TH * C): Y = B
*  SIN (TH * C)
```



A elipse

A PARÁBOLA

O tamanho da parábola depende do valor de uma variável, conhecida por T. As equações são:

X = T
Y = 2*T



A parábola

O valor de T pode variar de infinito a menos infinito; grande parte de uma parábola, contudo, pode ser vista como T variando entre + 2 e − 2. No programa, esses valores têm que ser postos em escala por um fator M para que a curva caiba na tela do micro.



A hipérbole

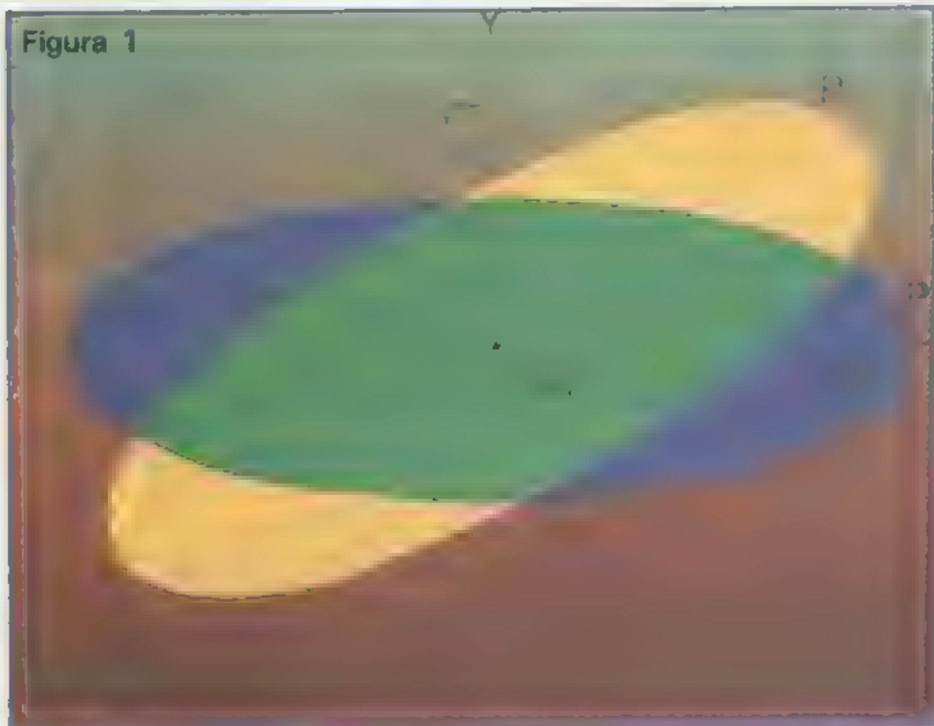


Figura 1

```
70 NEXT t
75 LET x=m/COS (PI 1)  LET y
m*TAN (PI 1)
80 PLOT 27+x,75+y
90 FOR t PI 1 TO PI+1 STEP
100 LET x m/COS t  LET y m*TAN
t
110 DRAW 27+x PEEK 23677,75+y
-PEEK 23678
120 NEXT t
```

```
10 PMODE 4:PCLS:SCREEN 1,1
15 M=50
20 C=ATN(1)/45
30 LINE -(227,8),PRESET
40 FOR TH=-60 TO 60 STEP 5
50 X=M/COS(TH*C):Y=M*TAN(TH*C)
60 LINE -(127+X,95+Y),PSET
70 NEXT TH
80 LINE-(26,8),PRESET
90 FOR TH=120 TO 240 STEP 5
100 X=M/COS(TH*C):Y=M*TAN(TH*C)
110 LINE-(127+X,95+Y),PSET
120 NEXT TH
130 GOTO 130
```

```
10 CLS
15 LET m=20
25 LET x=4*m: LET y 4*m
30 PLOT 127+x,80+y
40 FOR t=-2 TO 2 STEP .05
50 LET x=m*t*t: LET y=2*m*t
60 DRAW x PEEK 23677+127,y-
PEEK 23678+80
70 NEXT t
```

```
10 PMODE 4:PCLS:SCREEN 1,1
15 M=23
20 C=ATN(1)/45
30 LINE-(127+M*4,95-4*M),PRESET
40 FOR T=-2 TO 2 STEP .05
50 X=M*T^2:Y=2*M*T
60 LINE-(127+X,95+Y),PSET
70 NEXT T
80 GOTO 80
```

```
10 COLOR15,4,4:SCREEN2
15 M=23
20 C=ATN(1)/45
30 LINE -(127+M*4,95 M*4),4
40 FOR T=-2 TO 2 STEP .05
50 X=M*T^2:Y 2*M*T
60 LINE -(127+X,95+Y),15
70 NEXT
80 GOTO 80
```

```
10  HOME : HGR2 : HCOLOR= 3
15 M = 20
20 C = ATN (1) / 45
30  HPLOT 127 + M * 4,95 - 4 *
M
40  FOR T = - 2 TO 2 STEP .05
50 X = M * T ^ 2:Y = 2 * M * T
60  HPLOT TO 127 + X,95 + Y
70  NEXT
```

A HIPÉRBOLE

A equação da hipérbole é:

X = A/COS teta
Y = B*TG teta

Metade da hipérbole é traçada enquanto teta varia de -90° a 90° e a outra metade, enquanto teta varia de 90° a 270°. Teoricamente, é possível usar apenas um laço no programa para variar teta de -90 a +270, mas há problemas nos pontos -90, 90 e 270, onde ocorre uma divisão por zero. Mesmo com valores próximos destes, grandes números são envolvidos. Assim, o nosso programa recorre a dois laços. Novamente, o fator M vem estabelecer a escala mais adequada para a figura:

```
10 CLS
15 LET m=30
25 LET x=m/COS -1  LET y m*
TAN -1
30 PLOT 127+x,75+y
40 FOR t -1 TO 1 STEP .1
50 LET x=m/COS t  LET y=m*TAN
t
60 DRAW 127+x PEEK 23677,75+y
-PEEK 23678
```

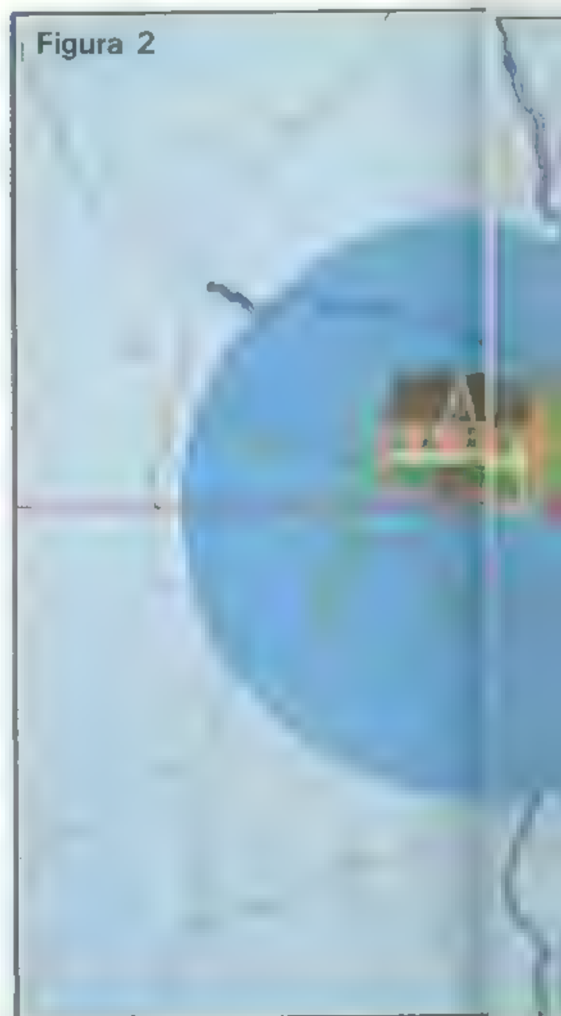


Figura 2



```
10 COLOR15,4,4:SCREEN2
15 M=50
20 C=ATN(1)/45
30 LINE (227,8),4
40 FOR TH=-60 TO 60 STEP 5
50 X=M/COS(TH*C):Y=M*TAN(TH*C)
60 LINE -(127+X,95+Y),15
70 NEXT
80 LINE (26,8),4
90 FOR TH=120 TO 240 STEP 5
100 X=M/COS(TH*C):Y=M*TAN(TH*C)
110 LINE -(127+X,95+Y),15
120 NEXT
130 GOTO 130
```



```
10  HOME : HGR2 : HCOLOR= 3
15  M = 50
20  C =  ATN (1) / 45
30   HPLOT 227,8
40   FOR TH = - 60 TO 60 STEP 5
50  X = M /  COS (TH * C):Y = M
 *  TAN (TH * C)
60   HPLOT  TO 127 + X,95 + Y
70   NEXT
80   HPLOT 26,8
90   FOR TH = 120 TO 240 STEP 5
100 X = M /  COS (TH * C):Y = M
 *  TAN (TH * C)
110  HPLOT  TO 127 + X,95 + Y
120  NEXT
```

COMO RODAR AS CURVAS

Os programas anteriores desenharam as curvas na sua forma mais simples, sobre um eixo horizontal e outro vertical. No entanto, essa posição nem sempre é conveniente; além disso, pode-se querer desenhar as curvas em outra posição. A figura 1 mostra o que acontece com um ponto na periferia de uma elipse quando esta sofre uma rotação de um ângulo de AN graus. O ponto P move-se da posição X,Y para uma nova posição XT, YT e as novas coordenadas são dadas por:

XT = X*COS AN-Y*SEN AN
YT = X*SEN AN + Y*COS AN



Eis aqui a rotina de rotação para cada computador:



```
1000 LET xt=x*COS (an*PI/180)-y
*SIN (an*PI/180)
1010 LET yt=x*SIN (an*PI/180)+y
*COS (an*PI/180)
1020 RETURN
```



```
1000 XT=X*COS(AN*C)-Y*SIN(AN*C)
1010 YT=Y*COS(AN*C)+X*SIN(AN*C)
1020 RETURN
```

Algumas alterações deverão ser feitas nos programas que desenham as curvas para que eles possam usar a sub-rotina de rotação. Em primeiro lugar, o ângulo de rotação tem que ser especificado (linha 17); a posição inicial deve ser rodada e as linhas, desenhadas nas novas coordenadas XT e YT. A linha 17 pode ser alterada para permitir a entrada do valor do ângulo para rotação por meio de uma instrução **INPUT**. É importante lembrar que essa instrução deve ser colocada antes da seleção do modo gráfico.

Vejamos agora quais são as mudanças que precisamos realizar no programa BASIC que desenha a elipse. Não se esqueça de incorporar a rotina de rotação a cada programa.



```
17 LET an=60
28 GOSUB 1000
30 PLOT 127+xt,70+yt
55 GOSUB 1000
60 DRAW xt-PEEK 23677+127,yt-
PEEK 23678+70
80 STOP
```



```
17 AN=60
25 X=A:GOSUB 1000
30 LINE-(127+XT,95+YT),PRESET
55 GOSUB 1000
60 LINE-(127+XT,95+YT),PSET
```



```
17 AN=60
25 X=A:GOSUB 1000
30 LINE -(127+XT,95+YT),4
55 GOSUB 1000
60 LINE -(XT+127,YT+95),15
```



```
17 AN = 60
25 X = A: GOSUB 1000
30  HPLOT 127 + XT,95 + YT
55  GOSUB 1000
60  HPLOT  TO 127 + XT,95 + YT
80  END
```

Para fazer a rotação da parábola deve-se empregar a mesma sub-rotina. Adicione-a ao programa principal e faça as seguintes alterações:

```
17 LET an=60
28 GOSUB 1000
30 PLOT 127+xt,80+yt
40 FOR t=-1.75 TO 1.75 STEP
.05
55 GOSUB 1000
60 DRAW 127+xt-PEEK 23677,80+
yt-PEEK 23678
80 STOP
```

```
17 AN=60
20 C=ATN(1)/45
25 X=M*4:Y=-M*4:GOSUB 1000
30 LINE -(127+XT,95+YT),PRESET
55 GOSUB 1000
60 LINE -(127+XT,95+YT),PSET
```

```
17 AN=60
25 X=M*4:Y=-M*4:GOSUB 1000
30 LINE -(127+XT,95+YT),4
55 GOSUB 1000
60 LINE -(XT+127,YT+95),15
```

```
15 M = 17
17 AN = 60
20 C =  ATN (1) / 45
25 X = M * 4:Y =  - M * 4: GOSU
B 1000
30  HPLOT 127 + XT,95 + YT
55  GOSUB 1000
60  HPLOT  TO 127 + XT,95 + YT
80  END
```

E, finalmente, as alterações para rodar a hipérbole:

```
17 LET an=60
28 GOSUB 1000
30 PLOT 127+xt,75+yt
55 GOSUB 1000
60 DRAW 127+xt-PEEK 23677,75+
yt-PEEK 23678
76 GOSUB 1000
80 PLOT 127+xt,75+yt
105 GOSUB 1000
110 DRAW 127+xt-PEEK 23677,75+
yt-PEEK 23678
130 STOP
```

```
17 AN=60
25 X=M/COS(-60*C):Y=M*TAN(-60*C
):GOSUB 1000
30 LINE -(127+XT,95+YT),PRESET
55 GOSUB 1000
60 LINE -(XT+127,YT+95),PSET
75 X=M/COS(135*C):Y=M*TAN(135*C
):GOSUB 1000
80 DRAW"BM"+STR$(INT(127+XT))+"
,"+STR$(INT(95+YT))
90 FOR TH=135 TO 240 STEP 5
105 GOSUB 1000
110 LINE -(127+XT,95+YT),PSET
```

```
17 AN=60
25 X=M/COS(-60*C):Y=M*TAN(-60*C
):GOSUB 1000
30 LINE -(127+XT,95+YT),4
55 GOSUB 1000
60 LINE -(XT+127,YT+95),15
75 X=M/COS(135*C):Y=M*TAN(135*C
):GOSUB 1000
80 LINE -(127+XY,95+YT),4
90 FOR TH=135 TO 240 STEP 5
105 GOSUB 1000
110 LINE -(XT+127,YT+95),15
```

```
15 M = 35
17 AN = 60
25 X = M /  COS (  - 60 * C): Y =
 M *  TAN ( - 60 * C): GOSUB 10
```


Figura 3

```
00
30  HPLOT 127 + XT,95 + YT
55  GOSUB 1000
60  HPLOT  TO 127 + XT,95 + YT
75 X = M /  COS (135 * C)·Y = M
 *  TAN (135 * C)· GOSUB 1000
80  HPLOT 127 + XT,95 + YT
90  FOR TH   135 TO 240 STEP 5
105  GOSUB 1000
110  HPLOT  TO 127 + XT,95 + YT

130  END
```


APLICAÇÕES PRÁTICAS

Todas essas curvas podem ser usadas de alguma forma prática.

O círculo tem tantas aplicações que nem todas podem ser listadas. A roda é um exemplo óbvio, assim como a bola também o é para a esfera. As esferas (ou formas aproximadas) aparecem com freqüência na natureza. Os exemplos vão desde laranjas até planetas. Mas elas raramente são perfeitas, devido à ação da gravidade, ventos ou outras forças. Um planeta rodando ao redor de uma estrela poderia ter uma órbita circular, apesar de ser mais provável que ela fosse elíptica.

As aplicações para o círculo são tão amplas como a determinação do menor custo de transporte para um produto que pode ser comprado em dois lugares diferentes. Suponhamos, por exemplo, que você queira comprar um modelo de computador vendido pelas firmas A e B, que estão distante 300 km entre si. Suponhamos ainda que a firma A envie o computador por um serviço especial de entrega a um custo de um cruzado por km, enquanto a firma B o faça em seu próprio caminhão, ao custo de cinqüenta centavos por km. É muito simples marcar num mapa a região em que é mais barato comprar de A ou de B. A idéia é unir por meio de uma linha todos os pontos onde os custos são iguais. Neste caso, devemos definir os lugares em que a distância até B seja o dobro da distância até A.

Um desses pontos fica na linha que liga as duas firmas, a 100 km de A e 200 km de B. Outro ponto fica na mesma linha, a 300 km de A, no sentido oposto a B. Se todos esses pontos forem unidos, teremos um círculo de raio igual a 200 km, como se vê na figura 2. Se você vive dentro do círculo, é mais barato comprar de A; se vive fora, é mais compensador comprar de B.

A elipse também tem aplicações práticas. É possível, por exemplo, colocar uma elipse em tal posição que sua sombra, projetada sobre uma superfície plana, assuma a forma de um círculo perfeito. Essa propriedade pode ser usada em válvulas de dutos circulares, onde uma aba elíptica é usada para controlar o fluxo de líquidos ou gases. Ao atingir determinado ângulo, a aba se encaixa perfeitamente no duto, bloqueando o fluxo da substância. A parábola descreve a curva traçada por um projétil. Mas certos corpos celestes, como os cometas, podem também viajar em órbitas parabólicas em torno do Sol e de outras estrelas.

MICRO DICAS

PARÁBOLAS E HIPÉRBOLES

Os programas deste artigo foram concebidos de modo a explorar da melhor maneira a tela do seu micro. Ao usar essas formas em programas, o fator de escala M deve ser alterado para que sejam obtidas curvas do tamanho adequado.

Ao rodar parábolas e hipérboles, é necessário tomar certos cuidados para que os pontos das extremidades das curvas não caiam fora da tela. Assim, altere o fim do laço **FOR...NEXT** da linha 40 do programa da parábola e das linhas 40 a 90 do programa da hipérbole. O limite exato deverá ser encontrado pelo método de tentativa e erro

Por outro lado, raios de luz e calor paralelos ao eixo de uma parábola são refletidos de modo a passar pelos focos. Isso acontece nas duas direções. Assim, um filamento colocado no foco produzirá um feixe paralelo de luz, como os usados em faróis de automóveis. Na outra direção, os raios solares podem ser concentrados no foco de uma parábola, produzindo altíssimas temperaturas; este é o princípio de funcionamento dos fornos solares.

Na prática, os refletores utilizados para tais fins são parabolóides tridimensionais. Estes também são usados em outras aplicações, como antenas para recepção e transmissão de ondas de rádio, acessórios de sistemas de televisão e telefonia etc.

Uma característica importante da hipérbole é — como já foi dito — sua constituição em duas partes. Os sistemas de radar para navegação valem-se dessa qualidade, funcionando com duas estações. Uma delas transmite um sinal, que é retransmitido imediatamente pela outra estação. Ao receber os dois sinais, o operador de um navio nas proximidades calcula o tempo entre as duas emissões. Se o barco se move de maneira a manter constante o tempo entre as chegadas, é porque tem um curso hiperbólico, como é mostrado na figura 3. Se o operador também receber sinais de outras duas estações e calcular o tempo entre a chegada dos sinais, terá outra hipérbole; a interseção das duas dará a posição do navio.

CRIE SPRITES COM VPEEK E VPOKE

Você pode criar e movimentar sprites com a ajuda dos versáteis comandos VPOKE e VPEEK. Mas, para isso, precisa conhecer primeiro a organização de memória de vídeo do MSX.

Até agora, usamos bastante os comandos **VPEEK** e **VPOKE** para obter efeitos gráficos, mas nunca na programação de sprites. Para fazer isso, precisamos conhecer bem a organização da memória de vídeo do MSX, a chamada VRAM. A estrutura dessa memória, bem como os princípios básicos dos comandos **VPEEK** e **VPOKE**, foi explicada no artigo *Os Comandos PEEK e POKE* (página 261).

A VRAM é uma memória independente, com capacidade de armazenar até 16 kbytes. Ela é utilizada por um *chip* especial, que se dedica integralmente ao controle da tela: o VDP — *Video Display Processor*.

O MSX pode dispor da tela de quatro formas: texto em quarenta colunas, texto em 32 colunas, gráficos em alta resolução e gráficos em baixa resolução. É o VDP que determina o tipo de tela que está sendo mostrado no vídeo. Quando estamos programando em BASIC, o comando **SCREEN** ajusta o VDP para mostrar o tipo de tela desejado.

Os diferentes tipos de tela nada mais são que maneiras diversas de interpretar os números que estão na VRAM. Para fazer essa interpretação, o VDP divide a memória de vídeo em regiões que são denominadas tabelas. Cada tabela tem sua função. No modo texto de quarenta colunas, por exemplo, existe uma tabela de nomes, cuja função consiste em armazenar os códigos ASCII das letras que aparecem no texto. Uma vez que coloquemos um código numa certa posição da tabela de nomes, o VDP utiliza esse valor a fim de encontrar os bytes que determinam o formato da letra em outra parte da VRAM (na chamada tabela de padrões).

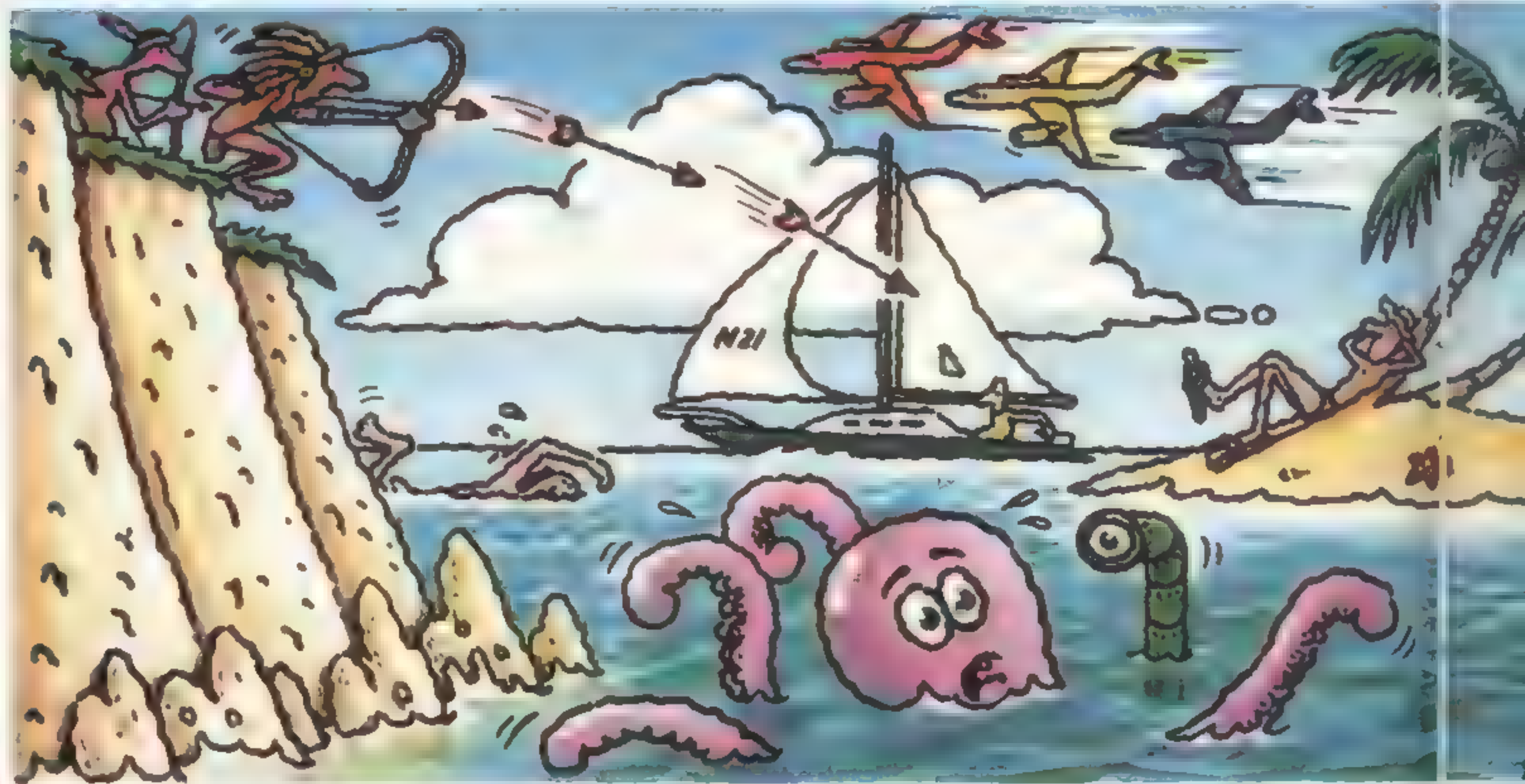
Os quatro tipos de tela têm tabelas de nomes e tabelas de padrões, que funcionam de maneira parecida. Para que não seja preciso memorizar os endereços iniciais de cada uma delas na VRAM, empregamos variáveis do sistema chamadas **BASE**, que descobrem o valor correto para nós. Esses valores estão contidos no VDP; em outro artigo aprenderemos a modificá-los.

No modo texto de quarenta colunas — **SCREEN 1** — não podemos modificar a cor de letras individuais nem usar sprites. Nos outros três tipos de tela existe uma porção da VRAM, dedicada a controlar os efeitos cromáticos, chamada tabela de cores. Essa tabela assume, em cada tela, um tamanho e uma posição diferentes.

Neste artigo, mostraremos a organização de duas outras tabelas da VRAM, existentes nos tipos de tela que permitem a programação de sprites: a tabela de atributos de sprites e a tabela de padrões de sprites. Mais adiante, trataremos da organização do VDP.

UM BANCO DE SPRITES

Antes de prosseguir no estudo da VRAM, convém recordar como o computador armazena o padrão de um spri-



- ORGANIZAÇÃO DA MEMÓRIA DE VÍDEO
- ONDE SÃO ARMAZENADOS OS PADRÕES DOS SPRITES
- COMO O COMPUTADOR MOVIMENTA SPRITES
- UM BANCO DE SPRITES EM LINHAS DATA
- CRIE SPRITES COM O EDITOR DE BLOCOS GRÁFICOS

te. Até agora, quando quisemos definir o pradrão de um sprite, usamos um cordão para armazenar os bytes correspondentes a cada linha com oito pontos do sprite. Se você consultar o artigo *Crie Sprites no MSX* (página 188), verá que a ordem de interpretação dos bytes tem a seguinte correspondência com o padrão de um sprite de 16 x 16 pontos:

Linha 1: BYTE 1 BYTE 17
Linha 2: BYTE 2 BYTE 18
Linha 3: BYTE 3 BYTE 19
Linha 4: BYTE 4 BYTE 20
. ... .

Para entender como os sprites são armazenados e movimentados na memória de vídeo, precisamos de alguns exemplos desses blocos gráficos. Neste artigo, como de costume, trataremos de sprites de 16 x 16 pontos, salvo menção em contrário. Digite o programa:

```
10 CLEAR 200,&HD000
20 FOR I 0 TO 255
30 READ A$:POKE &HD100+I,VAL("&
B"+A$)
40 POKE &HE100+I,16+15
50 NEXT
1000 DATA 00000000
1010 DATA 00000000
1020 DATA 00000000
1030 DATA 00000000
1040 DATA 00000000
1050 DATA 00000001
1060 DATA 00000011
1070 DATA 00111111
1080 DATA 11101010
1090 DATA 01111111
1100 DATA 00000001
1110 DATA 00000000
1120 DATA 00000000
1130 DATA 00000000
1140 DATA 00000000
1150 DATA 00000000
1160 DATA 00000000
1170 DATA 00000000
1180 DATA 00000000
1190 DATA 01110000
1200 DATA 10100000
1210 DATA 01000001
1220 DATA 10000001
1230 DATA 11111101
```



```
1240 DATA 10101001
1250 DATA 11111111
1260 DATA 00100000
1270 DATA 10010000
1280 DATA 01001000
1290 DATA 00111100
1300 DATA 00000000
1310 DATA 00000000
1320 DATA 00000000
1330 DATA 00000000
1340 DATA 00000000
1350 DATA 00111100
1360 DATA 00100010
1370 DATA 00100001
1380 DATA 01000001
1390 DATA 01000001
1400 DATA 10100010
1410 DATA 10000000
1420 DATA 11111101
1430 DATA 00110010
1440 DATA 00011101
1450 DATA 00000011
1460 DATA 00000000
1470 DATA 00000000
1480 DATA 00000000
1490 DATA 00000000
1500 DATA 00000000
1510 DATA 00000000
1520 DATA 00000000
1530 DATA 00000000
1540 DATA 11110000
1550 DATA 00101000
1560 DATA 00010100
1570 DATA 00101010
1580 DATA 01010101
1590 DATA 10111111
1600 DATA 01111100
1610 DATA 10000000
1620 DATA 00000000
1630 DATA 00000000
1640 DATA 00000000
1650 DATA 00000000
1660 DATA 00000000
1670 DATA 00000001
1680 DATA 00000001
1690 DATA 00000001
1700 DATA 11100001
1710 DATA 01010001
1720 DATA 00101010
1730 DATA 00100100
1740 DATA 00100000
1750 DATA 00101000
1760 DATA 01010100
1770 DATA 11100110
1780 DATA 00000101
1790 DATA 00000110
1800 DATA 00010010
1810 DATA 00000000
1820 DATA 00001001
1830 DATA 00000010
1840 DATA 10000000
1850 DATA 01000100
1860 DATA 00100001
1870 DATA 11111000
1880 DATA 00000100
1890 DATA 00110010
1900 DATA 00000001
1910 DATA 00111111
1920 DATA 01000000
1930 DATA 00111100
1940 DATA 11111000
1950 DATA 00000000
1960 DATA 00001110
1970 DATA 00000011
1980 DATA 00011011
1990 DATA 01101111
2000 DATA 10000111
2010 DATA 10011111
2020 DATA 00110011
2030 DATA 01100101
2040 DATA 01000101
2050 DATA 01000001
2060 DATA 00000001
2070 DATA 00000001
2080 DATA 00000001
2090 DATA 00000011
2100 DATA 00111111
2110 DATA 11111111
2120 DATA 00000000
2130 DATA 01111100
2140 DATA 11100110
2150 DATA 11111000
2160 DATA 11111100
2170 DATA 11100100
2180 DATA 11110010
2190 DATA 10011001
2200 DATA 10000100
2210 DATA 10000100
2220 DATA 10000000
2230 DATA 11000000
2240 DATA 11000000
2250 DATA 11000000
2260 DATA 11111100
2270 DATA 11111111
2280 DATA 00000111
2290 DATA 00111111
2300 DATA 00000000
2310 DATA 00000001
2320 DATA 00000010
2330 DATA 00000100
2340 DATA 00001000
2350 DATA 00010000
2360 DATA 00111111
2370 DATA 01111111
2380 DATA 00000000
2390 DATA 01111111
2400 DATA 00100000
2410 DATA 00011000
2420 DATA 00000111
2430 DATA 00000000
2440 DATA 10000000
2450 DATA 10000000
2460 DATA 10000000
2470 DATA 11000000
2480 DATA 11100000
2490 DATA 10110000
2500 DATA 10011000
2510 DATA 10000100
2520 DATA 10000010
2530 DATA 11111111
2540 DATA 10000000
2550 DATA 11111111
2560 DATA 00110010
2570 DATA 01100100
2580 DATA 11110000
2590 DATA 00000000
2600 DATA 00000000
2610 DATA 00000000
2620 DATA 00000000
2630 DATA 00000000
2640 DATA 00000000
2650 DATA 00000000
2660 DATA 00000000
2670 DATA 00000000
2680 DATA 00000000
2690 DATA 00000000
2700 DATA 00001100
2710 DATA 00011111
2720 DATA 00111111
2730 DATA 01111111
2740 DATA 11111111
2750 DATA 11111111
2760 DATA 00000000
```





```
2770 DATA 00000000
2780 DATA 00000000
2790 DATA 00000000
2800 DATA 00000000
2810 DATA 00000000
2820 DATA 00000000
2830 DATA 00010000
2840 DATA 00111000
2850 DATA 01111100
2860 DATA 11111100
2870 DATA 11111110
2880 DATA 11111110
2890 DATA 11111111
2900 DATA 11111111
2910 DATA 11111111
2920 DATA 00000000
2930 DATA 00000000
2940 DATA 00000000
2950 DATA 00000000
2960 DATA 00000011
2970 DATA 00000111
2980 DATA 00000110
2990 DATA 00000010
3000 DATA 00000011
3010 DATA 00000111
3020 DATA 00000111
3030 DATA 11111111
3040 DATA 00111111
3050 DATA 00001111
3060 DATA 00000000
3070 DATA 00000000
3080 DATA 00000000
3090 DATA 00000000
3100 DATA 00000000
3110 DATA 00000000
3120 DATA 00000000
3130 DATA 10000000
3140 DATA 00000000
3150 DATA 00000000
3160 DATA 00000000
3170 DATA 11000000
3180 DATA 01000000
3190 DATA 10111111
3200 DATA 11011100
3210 DATA 11100000
3220 DATA 00001000
3230 DATA 00000100
3240 DATA 00000111
3250 DATA 00010001
3260 DATA 00111101
3270 DATA 01001101
3280 DATA 01101101
3290 DATA 01101101
3300 DATA 01101101
3310 DATA 00101101
3320 DATA 00011101
3330 DATA 00001101
3340 DATA 00000110
3350 DATA 00000001
3360 DATA 00000111
3370 DATA 00000100
3380 DATA 00000100
3390 DATA 00000011
3400 DATA 11000000
3410 DATA 00010000
3420 DATA 01111000
3430 DATA 01100100
3440 DATA 01101100
3450 DATA 01101100
3460 DATA 01101100
3470 DATA 01101000
3480 DATA 01110000
3490 DATA 01100000
3500 DATA 11000000
3510 DATA 00000000
3520 DATA 11000000
3530 DATA 01000000
3540 DATA 01000000
3550 DATA 10000000
```

Esse programa cria um banco de sprites no topo da memória. Os formatos das figuras podem ser observados nos números binários contidos nas linhas **DATA**; basta lembrar que os "1" correspondem a pontos "acesos" e os "0", a pontos "apagados" na tela. Note a ordem dos bytes que compõem o padrão do sprite. Ela é exatamente aquela a que fizemos referência: primeiro os dezesseis bytes da metade esquerda e depois os da metade direita.

A linha 10 protege o topo da memória, onde coloca os padrões. O endereço usado é o mesmo dos bancos de blocos criados pelo editor dos artigos *Geração de Blocos Gráficos* (1 e 2).

Os laços **FOR...NEXT** entre as linhas 20 e 50 usam o comando **READ** para obter os números binários das linhas **DATA** e **POKE** para colocar os valores correspondentes no topo da memória. A linha 40 cuida das cores dos padrões — preto sobre fundo branco — para que possamos vê-los através do editor.

Essa utilização do editor de blocos para observar os padrões criados não é obrigatória. O BASIC aqui apresentado foi criado com objetivos didáticos: ele permite compreender a organização da VRAM com o auxílio do programa do final do artigo. As figuras geradas por ele — barcos, peixes e uma ilha deserta, entre outros — serão utilizadas em outro artigo para animar um quadro na tela do computador.

USE O EDITOR DE BLOCOS GRÁFICOS

Há dois caminhos para utilizar o editor de blocos gráficos com esse conjunto de sprites. O primeiro é mais indicado para principiantes. Depois de executar e gravar o programa acima em fita, grave o banco de sprites por intermédio do comando:

```
BSAVE'CAS:SPRITE',&HD100,&HEFFF
```

Carregue então o programa editor de blocos gráficos do cassete e execute-o. Aperte a tecla T e responda L à pergunta do computador: "(S)AVE ou (L)OAD?". Posicione a fita na posição em que gravou o banco de sprites, aperte **ENTER** e, em seguida, pressione o botão "PLAY" do gravador.

O segundo caminho é mais indicado para aqueles que estão acostumados com o uso do editor: depois de executar o programa apresentado linhas atrás, carregue o editor de blocos da fita e execute-o também. (O banco de sprites está na posição correta, só que não podemos vê-lo. É possível, contudo, recu-









perar os padrões com auxílio da tecla R.) As posições ocupadas pelos blocos vão de 0 a 31.

Os blocos de 8 x 8 pontos que compõem os sprites estão ordenados de acordo com as exigências do VDP.

SPRITES NA VRAM

Nos tipos de tela que permitem o uso de sprites, certas porções da VRAM são reservadas para controlar sua exibição. Para exemplificar, recorreremos à tela de 32 colunas.

O MSX só permite a exibição de 32 sprites ao mesmo tempo. Cada um desses sprites tem suas características armazenadas numa tabela de atributos de sprites. São quatro os atributos de um sprite: suas coordenadas X e Y, seu nome e sua cor. Dessa forma, cada sprite necessita de quatro bytes para seus atributos, e a tabela de atributos terá um comprimento de 128 bytes.

A prioridade de um sprite é determinada pela sua posição na tabela de atributos. Terão prioridade os sprites que ocuparem as primeiras posições na tabela; ou seja, se dois sprites estiverem na mesma posição, será mostrado aquele que aparecer primeiro na tabela. As coordenadas X e Y determinam o lugar do sprite na tela. Elas correspondem ao canto superior esquerdo do sprite. Seus valores devem permanecer entre 0 e 255. Quando Y é maior que 191, o sprite desaparece na borda inferior da tela. Se Y for 208, todos os sprites com prioridade inferior desaparecerão; e, se for 209, o sprite em questão desaparecerá.

É importante prestar atenção aos limites impostos à coordenada X. Quando utilizamos a instrução **PUT SPRITE**, a coordenada X pode ter valores negativos. Contudo, se tentarmos colocar em uma posição da VRAM um valor fora da faixa que vai de 0 a 255, o computador emitirá uma mensagem de erro.

Infelizmente, um sprite só pode ter uma cor. Esta corresponderá aos "pontos acesos" do padrão, ou à cor de frente, que é determinada pelos quatro bits menos significativos do byte de cor, na tabela de atributos. Os códigos das cores são aqueles que você já conhece. O bit 7 dessa posição controla os valores negativos de **X**, quando utilizados em uma instrução **PUT SPRITE**, ou seja, ele será "aceso" se X estiver entre 0 e −32. Essas coordenadas negativas permitem que o sprite desapareça no canto esquerdo da tela, podendo reaparecer no direito.

O nome do sprite indica ao VDP onde o padrão do sprite pode ser encontrado na tabela de padrões de sprites.

O endereço inicial da tabela de atributos de sprites está em **BASE(8)** na tela de 32 colunas; em **BASE(13)** na tela de alta resolução; e em **BASE(18)** na tela multicolor.

O MSX permite que criemos até 64 sprites de 16 x 16 pontos, embora só 32 possam aparecer na tela ao mesmo tempo. Os padrões desses sprites ficam guardados na tabela de padrões. Como cada sprite precisa de 32 bytes para definir seu padrão, a tabela tem 2048 bytes de comprimento. Os padrões definidos pelo BASIC **SPRITE$(N)** são colocados em posições diferentes dessa tabela, conforme o valor de N. O VDP se baseia no nome do sprite que está na tabela de atributos para calcular a posição em que seu formato se encontra na tabela de padrões.

Cada sprite de 16 x 16 pontos necessita, ao ser programado, de 32 bytes. Se preferirmos trabalhar com sprites pequenos, com 8 x 8 pontos, cada um deles precisará de apenas oito bytes para armazenar seu padrão. Dessa forma, como a tabela de padrões de sprites tem 2048 bytes de comprimento, podemos criar até 256 sprites pequenos e até 64 sprites grandes.

O endereço inicial da tabela de padrões de sprites fica em **BASE(9)** na tela de 32 colunas; em **BASE(14)** na tela de alta resolução; e em **BASE(19)** na tela multicolor.

Quando seu micro padrão MSX estiver ligado em um modo gráfico que permita a exibição de sprites, o VDP procurará na tabela de atributos de sprites os dados necessários à exibição dos 32 sprites ali definidos. De um certo modo, podemos dizer que o VDP está *sempre* mostrando os 32 sprites na tela; nem sempre, contudo, vemos todos os sprites — seja porque o valor da coordenada Y está escondendo o sprite, seja porque não há nenhum padrão definido na tabela de padrões e todos os pontos dos sprites estão apagados.

UM PROGRAMA DIDÁTICO

Para que funcione o programa a seguir — organizado, como o anterior, com propósitos didáticos —, é necessária a presença de um banco de sprites no topo da memória. Portanto, antes de digitá-lo, execute o programa anterior, apague-o com **NEW** e só então digite e execute a nova listagem.

```
5 CLEAR 200,&HD000
10 SCREEN 1,2:KEY OFF
20 COLOR 1,7,7:CLS
30 Z$="00000000":N=0:W=0
40 A=BASE(8):GOSUB 3000
50 VPOKE BASE(6)+4,16+9
55 VPOKE BASE(6)+31,12*16+12
60 X=VPEEK(A+4*N+1):Y=VPEEK(A+4
*N):C=VPEEK(A+4*N+3):NX=VPEEK(A
+N*4+2):P=BASE(9)+8*NX
```





```
70 GOSUB 1000:GOSUB 2000
80 K$=INKEY$:IF K$="" THEN 80
90 IF K$="N" AND N<31 THEN N=N+
1:GOTO 60
100 IF K$="B" AND N>0 THEN N=N-
1:GOTO60
110 IF K$="C" THEN C=(C+1)MOD16
:VPOKE A+4*N+3,C:GOTO 60
120 IF K$=CHR$(28) AND X<247 TH
EN X=X+8:VPOKE A+4*N+1,X:GOTO 6
0
130 IF K$=CHR$(29) AND X>7 THEN
 X=X-8:VPOKE A+4*N+1,X:GOTO 60
140 IF K$=CHR$(31) AND Y<247 TH
EN Y=Y+8:VPOKE A+4*N,Y:GOTO 60
150 IF K$=CHR$(30) AND Y>7 THEN
 Y=Y-8:VPOKE A+4*N,Y:GOTO 60
160 IF K$="L" THEN VPOKE A+4*N,
160:GOTO 60
170 IF K$="A" THEN VPOKE A+4*N,
209:GOTO 60
180 IFK$=CHR$(27) THEN W=NOT W:
GOTO 60
190 IF K$="W" AND NX<252THEN NX
=NX+4:VPOKE A+4*N+2,NX:GOTO 60
200 IF K$="Q" AND NX>3 THEN NX=
NX-4:VPOKE A+4*N+2,NX:GOTO 60
210 GOTO 80
1000 FOR I=0 TO 9
1010 LOCATE 0,I+13
1020 PRINT RIGHT$(STR$(P+I),5);
" ";RIGHT$(Z$+BIN$(VPEEK(P+I)),
8);
1030 NEXT I:IF W=-1 THEN 1090
1040 FOR I=10 TO 31
1050 LOCATE 14,I-9
1060 PRINT P+I;RIGHT$(Z$+BIN$(V
PEEK(P+I)),8);
1070 NEXT I
1080 RETURN
1090 FOR I=10 TO 31
1100 LOCATE 14,I-9
1110 PRINT STRING$(7,32);RIGHT$
(Z$+BIN$(VPEEK(P+I)),8);
1120 IF I=30 THEN I=I+1:GOTO 10
50
1130 NEXT I:RETURN
2000 LOCATE 3,1
2010 PRINT "SPRITE";N;
2020 LOCATE 1,3
2030 PRINT A+4*N;VPEEK(A+4*N);T
AB(12);"Y"
2040 LOCATE 1,4
2050 PRINT A+4*N+1;VPEEK(A+4*N+
1);TAB(12);"X"
2060 LOCATE 1,5
2070 PRINT A+4*N+2;VPEEK(A+4*N+
2)
2080 LOCATE 1,6
2090 PRINT A+4*N+3;" ";HEX$(VPE
EK(A+4*N+3))
2100 LOCATE 4,8
2110 PRINT "BASE(8)";
2120 LOCATE 4,9
2130 PRINT BASE(8)
2140 LOCATE 4,10
2150 PRINT "BASE(9)";
2160 LOCATE 4,11
2170 PRINT BASE(9)
2180 FOR I=0 TO 12
2190 VPOKE BASE(5)+I*32+2,255
2200 VPOKE BASE(5)+I*32+15,255
2210 NEXT
2220 FOR I=3 TO 14
2230 VPOKE BASE(5)+I,255
2240 VPOKE BASE(5)+I+32*12,255
2250 NEXT:RETURN
3000 FOR I=0 TO 255
3010 VPOKE BASE(9)+I,PEEK(&HD10
0+I)
3020 NEXT:RETURN
```

Ao ser executado, o programa mostra, no canto superior esquerdo da tela, cercado por uma moldura verde, um quadro onde se lê "SPRITE 0". Esse quadro é uma imagem da tabela de atributos de sprites. Nele são exibidos o endereço e o conteúdo das quatro posições da VRAM que correspondem ao primeiro sprite da tabela — o sprite 0. Quanto menor o número do sprite, maior será sua prioridade, o que dá ao sprite zero prioridade absoluta. Para que o leitor se localize na VRAM, os endereços iniciais da tabela de atributos de sprites — **BASE(8)** — e da tabela de padrões de sprites — **BASE(9)** — também são mostrados no quadro.

Como já dissemos, o primeiro byte corresponde à coordenada Y. Se olharmos a tela, entenderemos por que o sprite 0 — o avião — não está aparecendo: a sua coordenada Y é maior que 191. Para fazê-lo aparecer, basta pressionar L. Feito isso, surge instantaneamente na tela; se observarmos novamente o quadro de moldura verde, vamos entender por quê: a coordenada Y do sprite 0 mudou para 160.

O sprite pode ser movimentado com o auxílio das teclas do cursor. Note como as coordenadas vão se modificando com o movimento do sprite — a coordenada X é o segundo byte do quadro.

O terceiro byte mostrado no interior da moldura verde corresponde ao "nome" do sprite. Esse valor orienta o VDP na busca do padrão do sprite dentro da tabela de padrões.

O valor do nome, multiplicado por oito e somado ao endereço inicial da tabela de padrões — **BASE(9)** —, fornece o endereço do primeiro dos 32 bytes correspondentes ao padrão do sprite. São exatamente estes 32 bytes que estão sendo mostrados no restante da tela. Ali temos uma imagem da porção da VRAM que contém o padrão do sprite. Os endereços dos bytes estão em decimal, e o seu conteúdo, em binário. Veja como os "zeros" e os "uns" desenham exatamente o perfil do avião. Confira também o endereço inicial, multiplicando o nome por oito e somando o resultado ao conteúdo de **BASE(9)**.

Voltando ao quadro de moldura verde, resta o quarto byte, que corresponde à cor do sprite, em hexadecimal. Podemos mudar instantaneamente a cor do sprite pressionando a tecla C. Ligue o sprite 0 com a tecla L, movimente-o com as setas e veja como sua cor muda com a tecla C. Observe atentamente o conteúdo dos bytes da tabela de atributos enquanto a cor e a posição do sprite vão sendo modificadas.

Para ver o próximo sprite na ordem de prioridade, aperte a tecla N. Os quatro bytes seguintes da tabela de atributos de sprites aparecerão dentro da moldura verde e os 32 bytes correspondentes ao desenho do sprite, junto com seus endereços na tabela de padrões de sprites, aparecerão no resto da tela. A tecla L faz o sprite aparecer, e a tecla A o apaga da tela, colocando 209 no byte da coordenada Y.

Podemos então percorrer a tabela de atributos de sprites pressionando repetidas vezes a tecla N. A região de interesse dentro da tabela de padrões de sprites também é atualizada. Podemos ligar, movimentar e mudar as cores dos sprites. Depois de usar a tecla N 31 vezes, chegaremos ao fim da tabela de atributos. Para voltar, tomando a direção dos sprites com maior prioridade (ou menor número), basta pressionar a tecla B, que tem um efeito inverso ao da tecla N.

Podemos ainda modificar o nome de um sprite, alterando a região da tabela de padrões a que ele se refere. Isso equivale a trocar o padrão do sprite. Para avançar na tabela de padrões, mantendo a prioridade do sprite, use a tecla W. Se o sprite em questão estiver na tela, veremos seu padrão ser substituído por outro desenho. Para retroceder na tabela, use Q.

As teclas N e B não modificam nenhum valor da VRAM; elas apenas mudam a região da tabela de atributos de sprites mostrada dentro da moldura verde. Já as teclas Q e W modificam o byte do nome do sprite, alterando a figura que está sendo mostrada. Por exemplo, a tecla W transforma o avião em uma nuvem, e a Q coloca um coqueiro no lugar onde havia um barco.

Use o programa para entender a organização da VRAM. Quando não houver mais dúvidas sobre os significados de cada endereço, podemos lançar mão da tecla ESC. Ela apaga a coluna de endereços da direita.

O programa ajuda também a ilustrar uma limitação do uso de sprites: não é possível inserir em uma mesma linha horizontal mais do que quatro sprites. Tente colocar cinco na mesma linha horizontal e veja o que acontece.

COMO FUNCIONA O PROGRAMA

A linha 5 mantém protegida a área onde foi colocado o banco de sprites. A 10 seleciona a tela de 32 colunas e sprites grandes. A 20 seleciona as cores da tela, além de limpá-la.

As linhas 30 e 40 estabelecem os valores iniciais de algumas variáveis. **Z$** serve para imprimir números binários, **N** é a posição do sprite na tabela de atributos — ou seja, sua prioridade —, **W** é um sinalizador que indica se ESC foi pressionada, **A** contém o endereço inicial da tabela.

A mesma linha 40 chama a sub-rotina 3000, que é responsável pela transferência do banco de sprites para a tabela de padrões. As linhas 50 e 55 modificam as cores de dois grupos de caracteres. Os caracteres 32 e 255 são usados para desenhar respectivamente as molduras laranja e verde.

O laço principal do programa começa na linha 60. Ali são atualizados os valores de uma série de variáveis. **X** é a coordenada X do sprite atual; **Y** é a outra coordenada desse sprite; **C** é a sua cor e **NX**, o seu nome. Todos esses valores são obtidos na tabela de atributos de sprites por meio de comandos **VPEEK**. **P** é o endereço inicial da porção da tabela de padrões onde se encontra o desenho do sprite atual; seu valor é calculado somando-se ao endereço inicial da tabela de padrões o resultado da multiplicação de oito pelo nome do sprite atual.

A seguir, a linha 70 chama duas sub-rotinas. A que começa na linha 1000 é responsável pela impressão da imagem da tabela de padrões na tela. A da linha 2000, por sua vez, cuida da impressão da imagem da tabela de atributos, junto com sua moldura verde.

A linha 80 aguarda que pressionemos uma tecla e a série de instruções **IF... THEN** que se segue executa as funções correspondentes a cada tecla.

Caso a tecla N tenha sido pressionada, a linha 90 modificará o sprite atual, passando ao sprite seguinte — de menor prioridade. Para fazer isso, basta aumentar o valor da variável **N** em uma unidade. A linha 100 cuida da tecla **B**, que volta ao sprite anterior, subtraindo uma unidade de **N**.

A linha 110 modifica a cor do sprite quando pressionamos a tecla C. A nova cor é obtida somando-se uma unidade à variável **C**. A função **MOD16** é utilizada para manter o código da cor entre 0 e 15. **C+1 MOD16** é o resto da divisão do código da nova cor por 16. O código da nova cor é colocado na tabela de atributos por **VPOKE**.

As linhas 120 e 130 cuidam do movimento horizontal do sprite. Conforme as teclas "seta para a esquerda" e "seta para direita" são presssionadas, o valor da variável **X** aumenta ou diminui oito unidades. Note que as condições **AND X<247** e **X>7** evitam que o sprite ultrapasse os limites da tela. A nova coordenada X é colocada na tabela de atributos por **VPOKE**.

As linhas 140 e 150 cuidam do movimento vertical de maneira análoga. A linha 160 "liga" o sprite quando L é precionada. Isso é feito com **VPOKE**, que coloca o valor 160 na posição da tabela de atributos que corresponde à coordenada Y. A linha 170 apaga o sprite ao toque da tecla A. Para obtermos esse efeito, a coordenada Y é modificada para o valor 209, com **VPOKE**. A linha 180 detecta a tecla ESC, modificando o valor do sinalizador **W**.

As linhas 190 e 200 alteram o valor do byte que corresponde ao nome do sprite atual. A tecla Q diminui o nome em quatro unidades e a tecla W o aumenta na mesma medida. O nome deve ser sempre múltiplo de 4 quando se trata de sprites grandes. É que um padrão de sprite tem 32 bytes. Como o nome é multiplicado por oito para calcular a posição do padrão na tabela, se o aumentarmos em quatro unidades, a posição na tabela avançará 8 x 4 = 32 unidades, passando para o padrão seguinte. Se o usuário pressionar qualquer outra tecla diferente das mencionadas, a linha 210 retornará à linha 80.

A sub-rotina da linha 1000 cuida da impressão dos valores contidos na tabela de padrões. O laço **FOR...NEXT** entre as linhas 1000 e 1030 imprime os dez primeiros endereços de interesse no canto inferior esquerdo da tela.

A seguir, conforme o valor de **W**, são impressos os outros 22 endereços. Se **W** for 0, o laço entre as linhas 1040 e 1070 imprimirá tanto os endereços quanto seus conteúdos. Se **W** for 1, o laço entre as linhas 1090 e 1130 imprimirá somente os conteúdos, preenchendo o espaço antes ocupado pelo endereço com espaços cor-de-laranja.

Os endereços são impressos com números decimais; os conteúdos aparecem em binário. A função **BIN$** calcula o valor binário; a função **RIGHT$**, juntamente com a variável **Z$**, faz com que sejam impressos exatamente oito dígitos binários por endereço. A função **STRING$** serve para imprimir sete espaços cor-de-laranja. A sub-rotina da linha 2000 cuida do quadro com moldura verde e do seu conteúdo. Os dois laços **FOR...NEXT** das linhas entre 2180 e 2250 desenham a moldura, que é composta por caracteres de código ASCII 255. No restante da rotina, **LOCATE** é utilizada juntamente com **PRINT** para escrever os endereços, conteúdos e suas legendas nas posições adequadas.

Os valores dos quatro bytes do sprite atual são obtidos na tabela de atributos por meio do comando **VPEEK**.

BLOCOS GRÁFICOS EM AVALANCHE

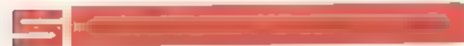
- JUNTE AS PORÇÕES DO VIDEOGAME
- DEFINA A FORMA DOS BLOCOS GRÁFICOS
- NÃO GRAVE BYTES A MAIS

A procura do lanche roubado leva nosso indigitado herói a enfrentar novos obstáculos. Agora, ele precisa de uma montanha para escalar. Use blocos gráficos para construí-la.

Que não seja o Everest, nem tampouco o Aconcagua. Nossa montanha há de ser modesta e simples: um mero acidente geográfico. Mas que represente um obstáculo a transpor, um desafio a vencer. Que Willie, ao vê-la, não resista ao impulso de escalá-la.

Para colocá-la na tela, precisamos criar uma série de blocos gráficos. Mais uma vez, usaremos um programa BASIC para colocar na memória a tabela com os padrões dos blocos. Não será possível (por enquanto) ver os gráficos na tela de alta resolução — a montagem do cenário é tarefa para uma rotina em código que será abordada no próximo artigo. Os usuários do Spectrum e do MSX, contudo, poderão ter uma idéia dos blocos, olhando o padrão dos bits nas linhas **DATA**. Neste estágio, tudo o que você deve fazer é digitar e executar o programa BASIC e gravar os códigos resultantes em fita.

A esta altura, temos vários pedaços de videogame gravados separadamente. Chegou o momento de juntá-los.



Precisamos, em primeiro lugar, definir o formato dos blocos gráficos utilizados no jogo — nuvens, gaivotas, pedras que rolam, buracos, cobras venenosas, lanche para um piquenique na relva, e o próprio Willie, nosso desventurado herói. Só esses caracteres já dariam muitos bytes, mas, para suavizar o processo de animação, todas as figuras móveis serão desenhadas em várias posições, o que aumenta ainda mais os dados. Além disso, empregaremos saltos de meio caractere e alternaremos os blocos para dar a impressão de movimento contínuo.

```
 5 CLEAR 56999
10 FOR n=57000 TO 57327: READ
a: LET a$=STR$ a: POKE n,VAL
("BIN"+a$): NEXT n
9010 DATA 00011000
9011 DATA 00111100
9012 DATA 00111100
9013 DATA 00011000
9014 DATA 00111100
9015 DATA 00111100
9016 DATA 00111100
9017 DATA 00111100
9018 DATA 00111100
9019 DATA 00111100
9020 DATA 00011000
```

```
9021 DATA 00011000
9022 DATA 00011000
9023 DATA 00011000
9024 DATA 00011000
9025 DATA 00011110
9026 DATA 00000001
9027 DATA 00000011
9028 DATA 00000011
9029 DATA 00000001
9030 DATA 00000000
9031 DATA 00000001
9032 DATA 00000001
9033 DATA 00000001
9034 DATA 10000000
9035 DATA 11000000
9036 DATA 11000000
9037 DATA 10000000
9038 DATA 00000000
9039 DATA 00000000
9040 DATA 00000000
9041 DATA 11100000
9042 DATA 00001110
9043 DATA 00000000
9044 DATA 00000001
9045 DATA 00000010
9046 DATA 00000100
9047 DATA 00001000
9048 DATA 00000100
9049 DATA 00000000
9050 DATA 00000000
9051 DATA 00000000
9052 DATA 10000000
9053 DATA 01000000
9054 DATA 00100000
9055 DATA 00100000
9056 DATA 00110000
9057 DATA 00000000
9058 DATA 00000100
9059 DATA 00000000
9060 DATA 00000000
9061 DATA 00000000
9062 DATA 00011000
9063 DATA 00111100
9064 DATA 00111100
9065 DATA 00011000
9066 DATA 00000000
9067 DATA 00010000
9068 DATA 00010000
9069 DATA 00011110
9070 DATA 11100000
9071 DATA 00000000
9072 DATA 00001100
9073 DATA 00100100
9074 DATA 01000010
9075 DATA 10000010
9076 DATA 01000011
9077 DATA 00000000
9078 DATA 00000000
9079 DATA 00000000
9080 DATA 00000000
9081 DATA 00000000
9082 DATA 00000000
9083 DATA 00000000
9084 DATA 00000000
9085 DATA 00000000
9086 DATA 00000001
9087 DATA 00000011
9088 DATA 00000011
9089 DATA 00000001
9090 DATA 00000000
9091 DATA 00000000
9092 DATA 00000000
9093 DATA 00000000
9094 DATA 10000000
9095 DATA 11000000
9096 DATA 11000000
9097 DATA 10000000
9098 DATA 00000000
9099 DATA 00000001
9100 DATA 00000001
9101 DATA 00000001
9102 DATA 00001110
9103 DATA 00000000
9104 DATA 00000001
9105 DATA 00000010
9106 DATA 00000000
9107 DATA 00000000
9108 DATA 00000000
9109 DATA 11100000
9110 DATA 00000000
9111 DATA 00000000
9112 DATA 10000000
9113 DATA 01000000
9114 DATA 00000100
9115 DATA 00001000
9116 DATA 00000100
9117 DATA 00000000
9118 DATA 00000000
9119 DATA 00000000
9120 DATA 00000000
9121 DATA 00000000
9122 DATA 00100000
9123 DATA 00100000
9124 DATA 00110000
9125 DATA 00000000
9126 DATA 00000000
9127 DATA 00000000
9128 DATA 00000000
9129 DATA 00000000
9130 DATA 00011100
9131 DATA 00111110
9132 DATA 01111111
9133 DATA 11111111
9134 DATA 11111111
9135 DATA 11111110
9136 DATA 11111100
9137 DATA 00111000
9138 DATA 00000011
9139 DATA 00000111
9140 DATA 00001111
9141 DATA 00001111
9142 DATA 00001111
9143 DATA 00000111
9144 DATA 00000011
9145 DATA 00000001
9146 DATA 10000000
9147 DATA 11000000
9148 DATA 11100000
9149 DATA 11110000
9150 DATA 11110000
9151 DATA 11110000
9152 DATA 11100000
9153 DATA 11000000
9154 DATA 00000000
9155 DATA 00000000
9156 DATA 00000111
9157 DATA 00011000
9158 DATA 00100000
9159 DATA 01000000
9160 DATA 01000000
9161 DATA 10000000
9162 DATA 00000000
9163 DATA 00000000
9164 DATA 00011111
9165 DATA 10100000
9166 DATA 11000000
9167 DATA 00000000
9168 DATA 00000000
9169 DATA 00000000
9170 DATA 00000000
9171 DATA 00000000
9172 DATA 10000000
9173 DATA 01000000
9174 DATA 01011100
9175 DATA 00100010
9176 DATA 00000010
9177 DATA 00000010
9178 DATA 10000000
9179 DATA 01000000
9180 DATA 01111100
9181 DATA 00000010
9182 DATA 00000010
9183 DATA 00000001
9184 DATA 00000000
9185 DATA 00000000
9186 DATA 00000000
9187 DATA 00000000
9188 DATA 00000000
9189 DATA 00000100
9190 DATA 00001010
9191 DATA 00010001
9192 DATA 11100000
9193 DATA 00000000
9194 DATA 00000010
9195 DATA 00000100
9196 DATA 00001000
9197 DATA 00000100
9198 DATA 00000100
9199 DATA 00000100
9200 DATA 11111000
9201 DATA 00000000
9202 DATA 00000000
9203 DATA 00000000
9204 DATA 01111000
9205 DATA 10000110
9206 DATA 00000001
9207 DATA 00000001
9208 DATA 00000000
9209 DATA 00000000
9210 DATA 00000000
9211 DATA 00000000
9212 DATA 00011110
9213 DATA 01100001
9214 DATA 10000000
9215 DATA 10000000
9216 DATA 00000000
9217 DATA 00000000
9218 DATA 00000000
9219 DATA 00000000
9220 DATA 00000000
9221 DATA 00000000
9222 DATA 10000111
9223 DATA 01111001
9224 DATA 00000000
9225 DATA 00000000
9226 DATA 00000000
9227 DATA 00000000
9228 DATA 00000000
9229 DATA 00000000
9230 DATA 11100001
9231 DATA 10011110
9232 DATA 00000000
9233 DATA 00000000
9234 DATA 00100010
9235 DATA 00010100
9236 DATA 00001000
9237 DATA 00001000
9238 DATA 00001000
9239 DATA 00001000
```

```
9240 DATA 00001000
9241 DATA 00001000
9242 DATA 00011000
9243 DATA 00111100
9244 DATA 00110110
9245 DATA 01111110
9246 DATA 01111110
9247 DATA 00111100
9248 DATA 00011000
9249 DATA 00011000
9250 DATA 00011000
9251 DATA 00011000
9252 DATA 00001100
9253 DATA 00001100
9254 DATA 00000110
9255 DATA 00000110
9256 DATA 00000011
9257 DATA 00000011
9258 DATA 00000110
9259 DATA 00000110
9260 DATA 00001100
9261 DATA 00001100
9262 DATA 00011000
9263 DATA 00011000
9264 DATA 00110000
9265 DATA 00110000
9266 DATA 01100000
9267 DATA 01100000
9268 DATA 11000110
9269 DATA 11000011
9270 DATA 01100110
9271 DATA 01101100
9272 DATA 00111000
9273 DATA 00111000
9274 DATA 10000100
9275 DATA 11010110
9276 DATA 11111111
9277 DATA 11111111
9278 DATA 11111111
9279 DATA 11111111
9280 DATA 11111111
9281 DATA 11111111
9282 DATA 00000000
9283 DATA 00000001
9284 DATA 00000011
9285 DATA 00000111
9286 DATA 00011111
9287 DATA 00111111
9288 DATA 01111111
9289 DATA 11111111
9290 DATA 00000000
9291 DATA 00000000
9292 DATA 11111111
9293 DATA 11111111
9294 DATA 00111100
9295 DATA 00111100
9296 DATA 11111111
9297 DATA 11111111
9298 DATA 00000110
9299 DATA 00001000
9300 DATA 01110110
9301 DATA 11111111
9302 DATA 11111111
9303 DATA 11111111
9304 DATA 01111110
9305 DATA 00111100
9306 DATA 00010000
9307 DATA 00010000
9308 DATA 00010000
9309 DATA 00111000
9310 DATA 00111000
9311 DATA 00111000
9312 DATA 00111000
9313 DATA 00111000
9314 DATA 00010000
9315 DATA 00111000
9316 DATA 01111100
9317 DATA 00111000
9318 DATA 00111000
9319 DATA 00111000
9320 DATA 00010000
9321 DATA 00010000
9322 DATA 00000000
9323 DATA 00000000
9324 DATA 00000000
9325 DATA 00100000
9326 DATA 01010001
9327 DATA 10001010
9328 DATA 00000100
9329 DATA 00000000
9330 DATA 00000000
9331 DATA 00000000
9332 DATA 00000000
9333 DATA 10000010
9334 DATA 01000101
9335 DATA 00101000
9336 DATA 00010000
9337 DATA 00000000
```

Execute o programa e grave a tabela resultante com o comando:

```
SAVE 'AVAL4' CODE 57000,327
```

JUNTE OS PEDAÇOS

Longos programas em código geralmente têm de ser montados por partes. Isso é bom para testar as rotinas individualmente, mas tem a desvantagem de deixar para o leitor a tarefa de juntar os pedaços do programa. O fato de as rotinas isoladas serem executadas com sucesso não garante o funcionamento do programa completo.

A maior dificuldade surge quando uma das partes apaga outra durante a montagem. Se você gravou mais bytes do que devia junto com um dos segmentos do programa, pode acontecer que eles apaguem parte do outro segmento adjacente na memória. Quando gravarmos os códigos de uma porção do jogo, usando o monitor de *INPUT*, devemos informar o endereço inicial e o número exato de bytes a transferir para a fita. Se utilizarmos o comando **SAVE** do Spectrum devemos recorrer à sintaxe:

```
SAVE 'nome' CODE
```

seguida do endereço inicial, uma vírgula e o número de bytes a gravar.

Quando usamos o Assembler para montar os programas listados nos artigos, o endereço inicial é sempre igual à origem: o número que vem após o falso mnemônico **org**. O próprio Assembler cuida de começar a rotina nesse endereço. Nos programas BASIC que criam tabelas de dados na memória a partir de linhas **DATA** e usando **POKE**, o endereço inicial da tabela pode ser encontrado na própria listagem: valor inicial da variável de controle do laço **FOR... NEXT** responsável pela leitura e transferência dos dados para a memória.

Calcular o número exato de bytes a serem gravados já é mais difícil. O endereço final informado pelo Assembler nem sempre é o do final da rotina. Muitas vezes aquele endereço é apenas o da última instrução montada, que, em várias listagens, é um rótulo seguido por um asterisco, usado para cálculo de saltos e desvios, e não correspondendo ao final do programa.

Certos programadores gostam, por segurança, de gravar alguns bytes a mais; eles correm o risco, porém, de apagar pedaços de outras rotinas no processo de montagem do jogo completo.

A maneira mais segura de juntar as partes do programa é ler as rotinas no gravador, preenchendo a memória em ordem crescente. Porções com endereços iniciais menores devem ser recuperadas primeiro, sendo seguidas sempre das rotinas que ocupam as posições sucessivas. Isso fará os eventuais bytes gravados em excesso serem apagados pela próxima rotina (e não o contrário). Além disso, qualquer instrução **ret**, colocada no final de uma rotina apenas para permitir o seu funcionamento isolado, também será apagada.

Quando todas as rotinas montadas tiverem sido colocadas na memória e estiverem funcionando adequadamente, o programa resultante deve ser gravado inteiro, com um outro nome. Seu endereço inicial será o da primeira rotina lida na fita e o endereço final será o endereço final da última rotina colocada na memória. Qualquer problema pode exigir que as rotinas sejam reunidas novamente. Não se esqueça de gravar as listagens Assembly — os programas-fonte —, bem como os programas BASIC que criam as tabelas na memória, pois você pode precisar repetir o processo de montagem.

Se houver qualquer problema com as tabelas de dados, podemos montar as rotinas em código na ordem citada, e executar cada um dos programas BASIC. Eles devem ser lidos, um a um, no gravador, executados, e, em seguida, apagados com **NEW**. Depois disso, o próximo programa criador de tabelas será lido na fita e o processo se repetirá.

De início, é preciso definir o formato dos blocos gráficos utilizados no jogo — nuvens, gaivotas, pedras que rolam, buracos, cobras venenosas, gulo-

seimas para um piquenique, e o próprio Willie. No MSX é mais fácil movimentar figuras usando sprites para imprimir suavidade à animação.

As figuras definidas pelo programa a seguir têm a estrutura de sprites. Nem todas, porém, serão usadas como tal. O MSX só permite a exibição simultânea de 32 sprites; destes, apenas quatro, no máximo, podem ocupar a mesma linha horizontal. Mas isso poderia fazer o pobre Willie desaparecer da tela. (Imagine se houvesse quatro serpentes na mesma linha horizontal!)

```
10 CLEAR 200,-15200
20 FOR I=0 TO 639
30 READ A$:POKE -15200+I,VAL("&
B"+A$)
40 NEXT
9000 DATA 00011000
9010 DATA 00111100
9020 DATA 00111100
9030 DATA 00011000
9040 DATA 00111100
9050 DATA 00111100
9060 DATA 00111100
9070 DATA 00111100
9080 DATA 00111100
9090 DATA 00111100
9100 DATA 00011000
9110 DATA 00011000
9120 DATA 00011000
9130 DATA 00011000
9140 DATA 00011000
9150 DATA 00011110
9160 DATA 0,0,0,0,0,0,0,0
9170 DATA 0,0,0,0,0,0,0,0
9180 DATA 00000001
9190 DATA 00000011
9200 DATA 00000011
9210 DATA 00000001
9220 DATA 00000000
9230 DATA 00000001
9240 DATA 00000001
9250 DATA 00000001
9260 DATA 00001110
9270 DATA 00000000
9280 DATA 00000001
9290 DATA 00000010
9300 DATA 00000100
9310 DATA 00001000
9320 DATA 00000100
9330 DATA 00000000
9340 DATA 10000000
9350 DATA 11000000
9360 DATA 11000000
9370 DATA 10000000
9380 DATA 00000000
9390 DATA 00000000
9400 DATA 00000000
9410 DATA 11100000
9420 DATA 00000000
9430 DATA 00000000
9440 DATA 10000000
9450 DATA 01000000
9460 DATA 0010000
9470 DATA 00100000
9480 DATA 00110000
9490 DATA 00000000
9500 DATA 0,0,0,0,0,0,0,0
9510 DATA 0,0,0,0,0,0,0,0
9520 DATA 00011000
9530 DATA 00111100
9540 DATA 00111100
9550 DATA 00011000
9560 DATA 00000000
9570 DATA 00010000
9580 DATA 00010000
9590 DATA 00011110
9600 DATA 11100000
9610 DATA 00000000
9620 DATA 00001100
9630 DATA 00100100
9640 DATA 01000010
9650 DATA 10000010
9660 DATA 01000011
9670 DATA 00000000
9680 DATA 0,0,0,0,0,0,0,0
9690 DATA 00011100
9700 DATA 00111110
9710 DATA 01111111
9720 DATA 11111111
9730 DATA 11111111
9740 DATA 11111110
9750 DATA 11111100
9760 DATA 00111000
9770 DATA 0,0,0,0,0,0,0,0
9780 DATA 0,0,0,0,0,0,0,0
9790 DATA 0,0,0,0,0,0,0,0
9800 DATA 00000011
9810 DATA 00000111
9820 DATA 00001111
9830 DATA 00001111
9840 DATA 00001111
9850 DATA 00000111
9860 DATA 00000011
9870 DATA 00000001
9880 DATA 0,0,0,0,0,0,0,0
9890 DATA 10000000
9900 DATA 11000000
9910 DATA 11100000
9920 DATA 11110000
9930 DATA 11110000
9940 DATA 11110000
9950 DATA 11100000
9960 DATA 11000000
9970  DATA 00000000
9980  DATA 00000000
9990  DATA 00000111
10000 DATA 00011000
10010 DATA 00100000
10020 DATA 01000000
10030 DATA 01000000
10040 DATA 10000000
10050 DATA 10000000
10060 DATA 01000000
10070 DATA 01111100
10080 DATA 00000010
10090 DATA 00000010
10100 DATA 00000001
10110 DATA 00000000
10120 DATA 00000000
10130 DATA 00000000
10140 DATA 00000000
10150 DATA 00011111
10160 DATA 10100000
10170 DATA 11000000
10180 DATA 00000000
10190 DATA 00000000
10200 DATA 00000000
10210 DATA 00000000
10220 DATA 00000000
10230 DATA 00000000
10240 DATA 00000100
10250 DATA 00001010
10260 DATA 00010001
10270 DATA 11100000
10280 DATA 00000000
10290 DATA 00000000
10300 DATA 00000000
10310 DATA 10000000
10320 DATA 01000000
10330 DATA 01011100
10340 DATA 00100010
10350 DATA 00000010
10360 DATA 00000010
10370 DATA 00000010
10380 DATA 00000100
10390 DATA 00001000
10400 DATA 00000100
10410 DATA 00000100
10420 DATA 00000100
10430 DATA 11111000
10440 DATA 00000000
10450 DATA 0,0,0,0,0,0,0,0
10460 DATA 0,0,0,0,0,0,0,0
10470 DATA 00000000
10480 DATA 00000000
10490 DATA 01111000
10500 DATA 10000110
10510 DATA 00000001
10520 DATA 00000001
10530 DATA 00000000
10540 DATA 00000000
10550 DATA 0,0,0,0,0,0,0,0
10560 DATA 00000000
10570 DATA 00000000
10580 DATA 00011110
10590 DATA 01100001
10600 DATA 10000000
10610 DATA 10000000
10620 DATA 00000000
10630 DATA 00000000
10640 DATA 0,0,0,0,0,0,0,0
10650 DATA 00000000
10660 DATA 00000000
10670 DATA 00000000
10680 DATA 00000000
10690 DATA 10000111
10700 DATA 01111001
10710 DATA 00000000
10720 DATA 00000000
10730 DATA 0,0,0,0,0,0,0,0
10740 DATA 00000000
10750 DATA 00000000
10760 DATA 00000000
10770 DATA 00000000
10780 DATA 11100001
10790 DATA 10011110
10800 DATA 00000000
10810 DATA 00000000
10820 DATA 0,0,0,0,0,0,0,0
10830 DATA 00100010
10840 DATA 00010100
10850 DATA 00001000
10860 DATA 00001000
10870 DATA 00001000
10880 DATA 00001000
10890 DATA 00001000
10900 DATA 00001000
10910 DATA 00011000
10920 DATA 00111100
10930 DATA 00110110
10940 DATA 01111110
10950 DATA 01111110
10960 DATA 00111100
10970 DATA 00011000
```

```
11120 DATA 00001100
11130 DATA 00011000
11140 DATA 00011000
11150 DATA 00110000
11160 DATA 00110000
11170 DATA 0,0,0,0,0,0,0,0
11180 DATA 0,0,0,0,0,0,0,0
11190 DATA 01100000
11200 DATA 01100000
11210 DATA 11000110
11220 DATA 11000011
11230 DATA 01100110
11240 DATA 01101100
11250 DATA 00111000
11260 DATA 00011000
11270 DATA 0,0,0,0,0,0,0,0
11280 DATA 0,0,0,0,0,0,0,0
11290 DATA 0,0,0,0,0,0,0,0
11300 DATA 10000100
11310 DATA 11010110
11320 DATA 11111111
11330 DATA 11111111
11340 DATA 11111111
11350 DATA 11111111
11360 DATA 11111111
11370 DATA 11111111
11380 DATA 0,0,0,0,0,0,0,0
11390 DATA 0,0,0,0,0,0,0,0
11400 DATA 0,0,0,0,0,0,0,0
11410 DATA 00000000
11420 DATA 00000001
11430 DATA 00000011
11440 DATA 00000111
11450 DATA 00011111
11460 DATA 00111111
11470 DATA 01111111
11480 DATA 11111111
11490 DATA 0,0,0,0,0,0,0,0
11500 DATA 0,0,0,0,0,0,0,0
11510 DATA 0,0,0,0,0,0,0,0
11520 DATA 00000000
11530 DATA 00000000
11540 DATA 11111111
11550 DATA 11111111
11560 DATA 00111100
11570 DATA 00111100
11580 DATA 11111111
11590 DATA 11111111
11600 DATA 0,0,0,0,0,0,0,0
11610 DATA 0,0,0,0,0,0,0,0
11620 DATA 0,0,0,0,0,0,0,0
11630 DATA 00000110
11640 DATA 00001000
11650 DATA 01110110
11660 DATA 11111111
11670 DATA 11111111
11680 DATA 11111111
11690 DATA 01111110
11700 DATA 00111100
10980 DATA 00011000
10990 DATA 0,0,0,0,0,0,0,0
11000 DATA 0,0,0,0,0,0,0,0
11010 DATA 00011000
11020 DATA 00011000
11030 DATA 00001100
11040 DATA 00001100
11050 DATA 00000110
11060 DATA 00000110
11070 DATA 00000011
11080 DATA 00000011
11090 DATA 00000110
11100 DATA 00000110
11110 DATA 00001100
11710 DATA 0,0,0,0,0,0,0,0
11720 DATA 0,0,0,0,0,0,0,0
11730 DATA 0,0,0,0,0,0,0,0
11740 DATA 00010000
11750 DATA 00010000
11760 DATA 00010000
11770 DATA 00111000
11780 DATA 00111000
11790 DATA 00111000
11800 DATA 00111000
11810 DATA 00111000
11820 DATA 0,0,0,0,0,0,0,0
11830 DATA 0,0,0,0,0,0,0,0
11840 DATA 0,0,0,0,0,0,0,0
11850 DATA 00010000
11860 DATA 00111000
11870 DATA 01111100
11880 DATA 00111000
11890 DATA 00111000
11900 DATA 00111000
11910 DATA 00010000
11920 DATA 00010000
11930 DATA 0,0,0,0,0,0,0,0
11940 DATA 0,0,0,0,0,0,0,0
11950 DATA 0,0,0,0,0,0,0,0
11960 DATA 00000000
11970 DATA 00000000
11980 DATA 00000000
11990 DATA 00100000
12000 DATA 01010001
12010 DATA 10001010
12020 DATA 00000100
12030 DATA 00000000
12040 DATA 0,0,0,0,0,0,0,0
12050 DATA 0,0,0,0,0,0,0,0
12060 DATA 0,0,0,0,0,0,0,0
12070 DATA 00000000
12080 DATA 00000000
12090 DATA 00000000
12100 DATA 10000010
12110 DATA 01000101
12120 DATA 00101000
12130 DATA 00010000
12140 DATA 00000000
12150 DATA 0,0,0,0,0,0,0,0
12160 DATA 0,0,0,0,0,0,0,0
12170 DATA 0,0,0,0,0,0,0,0
```

Execute o programa e grave a tabela resultante com o comando:

```
BSAVE 'CAS:AVAL' -15100,-14501
```

AS PARTES E O TODO

Longos programas em código devem ser montados por partes. Bom para testar as rotinas individualmente, esse método tem, contudo, a desvantagem de deixar para o leitor a tarefa de juntar os pedaços do programa. O fato de as rotinas isoladas serem executadas com sucesso não garante o funcionamento do programa completo.

Como no exemplo anterior, a maior dificuldade consiste em uma das partes ser apagada por outra durante o processo de montagem. Se você gravou mais bytes do que devia junto com um dos segmentos do programa, corre o risco de ver alguns desses bytes apagarem parte de outro pedaço, adjacente na memória. Assim, quando gravarmos os códigos de uma porção do jogo usando o monitor de *INPUT*, devemos informar o endereço inicial e o endereço final para determinar o número exato de bytes a serem transferidos para a fita. Se recorrermos ao comando **BSAVE** do MSX, devemos usar a sintaxe:

```
BSAVE 'CAS:NOME'
```

seguida do endereço inicial, uma vírgula e o endereço final da porção a ser gravada.

Quando usamos o Assembler para montar os programas dos artigos, o endereço inicial é sempre igual a origem (ou seja, o número que vem após o falso mnemônico **org**). O próprio Assembler cuida de começar a rotina nesse endereço. Nos programas BASIC que criam tabelas de dados na memória a partir de linhas **DATA** e usando **POKE**, o endereço inicial da tabela pode ser encontrado na própria listagem — valor inicial da variável de controle do laço **FOR...NEXT** responsável pela leitura e transferência dos dados para a memória.

Já o cálculo do número de bytes a gravar é mais difícil, pois o endereço final informado pelo Assembler nem sempre é o do final da rotina.

Há quem prefira gravar alguns bytes a mais, por segurança; nesse caso, porém, corre-se o risco de apagar pedaços de outras rotinas no processo de montagem do jogo completo.

O modo mais seguro de juntar as partes do programa consiste em ler as rotinas no gravador, preenchendo a memória em ordem crescente. Porções com endereços iniciais menores devem ser recuperadas da fita primeiro, sendo seguidas sempre das rotinas que ocupam posições sucessivas. Isso fará os bytes gravados em excesso serem apagados pela próxima rotina (e não o contrário). Assim, qualquer instrução **ret**, colocada no final de uma rotina apenas para permitir o seu funcionamento isolado, também será apagada.

Quando todas as rotinas montadas tiverem sido colocadas na memória e estiverem funcionando, o programa resultante deve ser gravado inteiro, com outro nome. Seu endereço inicial será o da primeira rotina lida na fita e o final será o endereço final da última rotina colocada na memória. Qualquer problema pode exigir que as rotinas sejam reunidas novamente. Não se esqueça de gravar as listagens Assembly — os programas-fonte — bem como os programas BASIC que criam as tabelas na memória, pois você pode precisar repetir o processo de montagem.

Se houver qualquer tipo de problema com as tabelas de dados, podemos montar as rotinas em código, na ordem citada, e executar cada um dos programas em BASIC. Eles devem ser lidos no gravador, um a um, postos em execução e, em seguida, apagados pelo comando **NEW**. Depois disso, o próximo programa criador de tabelas será lido na fita e o processo se repetirá.



Adicione as próximas linhas ao programa criador de tabelas que vem sendo montado ao longo destes artigos:

```
110 READ A$
120 FOR A=1 TO LEN (A$)
130 POKE AD,ASC(MID$(A$,A,1))
140 AD=AD+1
150 NEXT A
160 DATA ###'##'###'##!####!##!
##!###'##''
170 FOR A=1 TO 702
180 READ A$:POKE AD,VAL("&H"+A$
)
190 AD=AD+1:NEXT A
200 IF AD<>18238 THEN PRINT "ER
ROR"
210 DATA 55,54,50,50,40,40,0,0,
7F,5F,57,D7,F5,FF,D5,75,DD,77,5
D,D5,7D,75,5D,77,F5,FD,57,75,DD
,77,5D,D5,FD,5F,57,D7,5D,FF,D5,
7F,75,77,FD,F7,D5,5D,75,77,55,D
5,D5,5D,5D,D7,F5,7D,D5,5D,5D,D7
,55,57,FF,7F,57,57,FD,FD
220 REM sol
230 DATA 55,75,D5,55,55,75,D5,5
5,75,75,D5,D5,5D,5D,D5,D5,5D,5D
,D7,55,57,5D,D7,5D,D5,D5,5D,5D,
75,D7,DD,75,7D,5D,75,D5,5D,75,5
D,D7,57,75,5D,5D,F5,D5,57,75,5D
,D5,57,75,57,55,55,D7,F7,55,55,
DD
240 DATA 57,55,55,D5,77,55,55,D
F,D5,D5,57,55,5D,D5,57,7F,F5,75
,5D,55,57,75,5D,F5,57,5D,75,5F,
5D,57,D7,55,75,75,57,55,55,77,7
5,D5,55,D7,75,D5,55,D7,75,75,57
,57,5D,75,57,57,5D,55,55,57,5D,
55
250 REM numeros
260 DATA 7D,D7,D7,D7,7D,5D,7D,5
D,5D,FF,7D,D7,5D,75,FF,FD,57,FD
,57,FD,5D,7D,DD,FF,5D,FF,D5,7D,
57,FD,7F,D5,FD,D7,7D,FF,57,5D,7
5,75,7D,D7,7D,D7,7D,7F,D7,7F,57
,57
270 REM graficos
280 DATA 57,5F,5F,57,5F,5F,5F,5
F,D5,F5,F5,D5,F5,F5,F5,F5,5F,5F
,57,57,57,57,57,57,F5,F5,D5,D5,
D5,D5,D5,FD,55,55,55,55,55,55,5
5,55,57,5F,5F,57,55,57,57,57,D5
,F5,F5,D5,55,55,55,FD,55,55,55,
55,55,55,55,55,55,55,55,55,55,5
5,55,55,FD,55,57,5D,75,D5,75,55
290 DATA 55,55,D5,75,5D,5D,5F,5
5,55,55,55,55,55,55,55,55,55,55
,55,55,57,5F,5F,57,55,55,55,55,
D5,F5,F5,D5,55,57,57,57,FD,55,5
7,5D,55,55,55,FD,55,55,D5,75,75
,D5,75,55,55,55,55,55,5D,5D,5F,
55,55,55,55,55,55,55,55,55,55,5
5,55,55,55,55,55,55,57,5F,5F,57
300 DATA 55,55,55,55,D5,F5,F5,D
5,55,55,55,55,55,55,55,55,55,55
,55,55,55,55,55,55,55,57,57,57,
FD,55,57,5D,55,55,55,FD,55,55,D
5,75,55,55,55,55,55,55,55,55,55
,55,55,55,55,55,55,55,75,D5,75,
55,55,55,55,55,5D,5D,5F,55,55,5
5,55,55,55,55
310 DATA 55,55,55,55,55,55,57,5
F,7F,FF,FF,FF,7F,5F,F5,FD,FF,FF
,FF,FD,F5,D5,55,55,55,55,55,55,
55,55,5F,7F,FF,FF,FF,7F,5F,57,D
5,F5,FD,FF,FF,FF,FD,F5,55,55,55
,55,55,55,55,55,5D,57,55,55,55,
55,55,55,5D,75,D5,D5,D5,D5,D5,D
5,57,5F,7D,7F,7F,5F,57,57,D5,F5
320 DATA FD,FD,FD,F5,D5,D5,57,5
7,55,55,55,55,55,55,D5,D5,F5,F5
,7D,7D,5F,5F,55,55,55,55,57,57,
5F,5F,7D,7D,F5,F5,D5,D5,55,55,7
D,7D,F5,F5,7D,7D,5F,5F,55,55,7D
,5F,7D,F5,D5,D5,55,55,AA,AA,56,
56,AA,AA,55,55,AA,AA,95,95,AA,A
A,55,55,7F,FF,FF,FF,7F,5F,81
330 DATA 15,7D,FF,FF,FF,FD,F5,5
7,57,57,5F,5F,5F,5F,5F,55,55,55
,D5,D5,D5,D5,57,5F,7F,5F,5F,5F,
57,57,55,D5,F5,D5,D5,D5,55,55,A
A,AA,AA,A6,99,6A,AA,AA,AA,AA,AA
,AA,A9,66,9A,AA,AA,AA,AA,6A,9A,
A6,A9,AA,AA,AA,AA,A6,99,6A,AA,A
A,
```

Os sustenidos e pontos de exclamação na linha 160 definem a silhueta da montanha. Cada sustenido corresponde a um caractere representativo de uma parte plana e cada ponto de exclamação, a um caractere de declive.

As linhas 210 a 330 contêm os bytes correspondentes aos padrões dos blocos gráficos utilizados — o sol, as pedras que rolam, as cobras venenosas e o próprio Willie. Pode parecer que há um excesso de dados na listagem, mas, para dar suavidade ao processo de animação, as figuras móveis serão desenhadas em posições diferentes.

Longos programas em código devem ser montados por partes. Isso é bom para testar as rotinas individualmente. Mas o sucesso na execução de rotinas isoladas não garante o funcionamento do programa completo. O principal perigo, neste como em outros casos, consiste em uma das partes apagar outra durante o processo de montagem. Assim, se você gravou mais bytes do que devia junto com um dos segmentos do programa, eles podem suprimir parte do outro pedaço adjacente na memória.

Quando gravarmos os códigos de uma porção do jogo usando o monitor de *INPUT*, devemos informar os endereços inicial e final, de modo a transferir para a fita o número exato de bytes. Se usarmos o comando **CSAVEM** do TRS-Color, devemos recorrer à sintaxe:

```
CSAVEM 'NOME',
```

seguida do endereço inicial, uma vírgula e o endereço final da porção a gravar. Quando usarmos o Assembler para montar os programas listados nos artigos, o endereço inicial será igual à origem (ou seja, o número que vem após o falso mnemônico **ORG**). O próprio Assembler cuidará de começar a rotina nesse endereço. Nos programas BASIC que criam tabelas de dados na memória a partir de linhas **DATA** e usando **POKE**, o endereço inicial da tabela pode ser encontrado na própria listagem — valor inicial da variável de controle do laço **FOR...NEXT** responsável pela leitura e transferência dos dados para a memória.

Mais difícil é a tarefa de calcular o número de bytes a serem gravados. É que o endereço final informado pelo Assembler nem sempre é o do final da rotina. Muitas vezes, aquele endereço é apenas o da última instrução montada, o que em muitas das nossas listagens significa um rótulo seguido por um asterisco, usado para cálculo de saltos e desvios, não correspondendo ao final do programa.

Procure não gravar bytes a mais para não apagar pedaços de outras rotinas no processo de montagem do jogo. A maneira mais segura de juntar as partes do programa consiste em ler as rotinas no gravador, preenchendo a memória em ordem crescente. Porções com endereços iniciais menores devem ser recuperadas da fita em primeiro lugar, sendo seguidas sempre das rotinas que ocupam posições sucessivas. Isso fará os bytes gravados em excesso serem apagados pela próxima rotina (e não o contrário). Assim, qualquer instrução **RTS**, colocada no final de uma rotina para permitir o seu funcionamento isolado, também será apagada.

Quando todas as rotinas montadas tiverem sido colocadas na memória e estiverem funcionando adequadamente, chegou o momento de gravar, inteiro, o programa resultante, com um outro nome. Seu endereço inicial será o da primeira rotina lida na fita e o endereço final será o endereço final da última rotina colocada na memória. Qualquer problema pode exigir que as rotinas sejam agrupadas novamente. Não se esqueça de gravas as listagens Assembly — os programas-fonte —, assim como os programas BASIC que criam as tabelas na memória, pois você pode precisar repetir o processo de montagem.

MÓDULO LUNAR: COMANDE O POUSO

- PAISAGEM LUNAR
- MARCADOR DE COMBUSTÍVEL
- VELOCÍMETRO
- CONTROLES DE POUSO

Você vai precisar de muita habilidade e sangue-frio para fazer uma nave pousar na superfície lunar em segurança. O jogo é envolvente e facílimo de programar. Confira!

[illegible] de longas e complicadas linhas. Como você verá neste artigo, podemos criar jogos muito interessantes [illegible] curtos.

Nosso jogo, Módulo Lunar, [illegible] completo. Ele oferece ao jogador gráficos de alta resolução e controle total sobre o módulo. Se você quiser acrescentar-lhe novas mensagens, como "Quer jogar novamente?", ou modifique a paisagem lunar, [illegible] pende de sua preferência.

CONTROLES

[illegible] sentido (esquerda/direita) de seu movimento, usando os foguetes para diminuir a velocidade de [illegible] plataforma lunar.

Os controles são [illegible] A e F para o Apple, o TK-2000 e o MSX; 6, 7 e 8 para o Spectrum [illegible] TRS-Color.

```
10 COLOR [illegible]:SCREEN2
20 FORN=1 TO 50:PSET(RND([illegible]
 RND(1)*140),15:NEXTN
30 DRAW"C6BM0,188M+10,[illegible]
15M+18,+8M+18,-8M+16,-20M+16,-[illegible]
M+13,+20M+14,+8M+18,+4M+10,0M+1
0,-10M+20,-25M+10,+20M+10,+10M+
20,+5M+10,-20M255,191"
40 PAINT(10,191),6
50 DRAW"C1BM145,180[illegible]
M+2,+4"
60 DRAW"BM93,157C1D31R1U31"
90 S=-TIME:LX=INT(RND(S)*240)[illegible]
0:LY=INT(RND(S)*10)+15:XV=INT(R
ND(S)*10):YV=0:F=[illegible]
100 AX=LX:AY=LY
110 GOSUB 4000
120 GOSUB 1000:[illegible]
 3000
130 [illegible]
```

```
[illegible] TO 200:NEXT
150 CLS:SCREEN0:COLOR 1,15,15
160 IF LX<147 OR LX>[illegible] THEN [illegible]
170 IF ABS([illegible])>4 THEN 200
180 LOCATE5,10:PRINT"parabéns, [illegible]
pouso bem sucedido":GOTO 210
190 LOCATE7,10:PRINT"[illegible] fora da
plataforma[illegible]":GOTO 210
200 LOCATE0,10:PRINT"nave danif
icada, velocidade [illegible]"
210 END
1000 IF LY>1 THEN GOSUB 4000
1010 LX=LX+XV:LY=LY+YV
1020 IF LX<5 THEN LX=LX+245
1030 IF LX>250 THEN LX=LX-245
1040 IF LY<1 THEN RETURN
1050 IF LY>150 THEN RETURN
1060 GOSUB 4000
1070 RETURN
2000 YV=YV+[illegible] THEN RETUR
N
2010 [illegible]
```



```
[illegible]
100 AX = LX:AY = LY
115 GOSUB 4000
[illegible]
170 FOR TT = 0 TO 2500: NEXT
[illegible]
1010 LX = LX + XV:LY = LY + YV
[illegible]
1037 IF LY > 150 THEN RETURN
[illegible]
2020 IF I$ = "A" THEN XV = XV
[illegible] RETURN
2030 IF I$ = "S" THEN XV = XV
[illegible]
4010 HPLOT AX,AY: HPLOT TO AX
 [illegible],AY + 10: HPLOT TO AX,AY [illegible]
[illegible]
```



```
-247*(LX>0)
1040 GET(LX,LY)-(LX+7,LY+7),B,G
1050 PUT(LX,LY)-(LX+7,LY+7),L,P
SET:RETURN
2000 YV=YV+.5:IF F<1 THEN RETUR
N
2010 IF PEEK(341)=247 AND F>3 T
HEN YV=YV-1:F=F-3:RETURN
2020 IF PEEK(343)=247 THEN XV=X
V-.5:F=F-1:RETURN
2030 IF PEEK(344)=247 THEN XV=X
V+.5:F=F-1
2040 RETURN
3000 LINE (9+F,1)-(12+F,5),PRES
ET,BF
3010 V=2*YV:IF ABS(V)>122 THEN
V=122*SGN(V)
3020 LINE(8,8)-(255,11),PRESET,
BF:LINE(132,8)-(132+V,11),PSET,
BF
3030 RETURN
```

S

```
10 BORDER 1: INK 7: PAPER 0:
 CLS : BRIGHT 1
20 FOR N=1 TO 50: PLOT RND*
255,(RND*135)+40: NEXT N
70 PLOT 0,0: FOR N=1 TO 16:
READ GX,GY: DRAW GX,GY: NEXT
N
80 DATA 18,30,18,-15,18,-8,18
,8,16,20,16,5,13,-20,16,-8,18
,-4,15,0,10,10,20,25,10,-20,
10,-10,20,-5,18,20
90 PRINT AT 0,4; INK 6; PAPER
 2;"COMB:";AT 0,18;"VELOCID :"
110 LET LX=RND*240+10: LET LY=
160-(15+(RND*10)): LET XV=RND*
15: LET YV=0: LET F=246
115 GOSUB 4000
120 GOSUB 1000: GOSUB 2000:
GOSUB 3000
130 IF LY>20 THEN  GOTO 120
135 PAUSE 50
140 CLS : IF LX<154 OR LX>164
OR ABS YV>4 THEN  GOTO 160
150 PRINT " PARABENS , POUSO B
EM SUCEDIDO!": RESTORE 5000:
FOR N=1 TO 14: READ A,B: SOUND
 A,B: NEXT N: GOTO 170
160 PRINT AT 10,7; FLASH 1;
INK 2; PAPER 7;"!!!!!CRASH!!!!
!": FOR T=1 TO 50: BORDER RND*
7: SOUND .01,RND*5: NEXT T
170 PAUSE 400
180 STOP
1000 IF LY<160 THEN  GOSUB 4000
1010 LET LX=LX+XV: LET LY=LY+YV
: IF LY<300 THEN  SOUND .02,LY/
5
1030 IF LX<5 THEN  LET LX=LX+24
5
1035 IF LX>250 THEN  LET LX=LX-
242
1036 IF LY>160 THEN  RETURN
1037 IF LY<10 THEN  RETURN
1040 GOSUB 4000
1050 RETURN
2000 LET YV=YV-.5: IF F<1 THEN
 RETURN
2010 IF INKEY$="7" AND F>3 THEN
  LET YV=YV+1: LET F=F-3: RETUR
N
2020 IF INKEY$="5" THEN  LET XV
=XV-.5: LET F=F-1: RETURN
2030 IF INKEY$="8" THEN  LET XV
=XV+.5: LET F=F-1: RETURN
2040 RETURN
3000 PRINT AT 0,10;" "+STR$ F+"
 ";AT 0,28;" "+STR$ INT YV+" "
3010 RETURN
4000 OVER 1: PLOT LX,LY: DRAW -
5,-10: DRAW 5,2: DRAW 5,-2: DRA
W -4,10
4010 OVER 0: RETURN
5000 DATA .2,4,.2,7,.2,5,.2,12,
.2,0,.2,4,.2,4,.2,5,.6,7,.2,12,
.2,0,.2,4,.2,2,.6,0
```

AVALANCHE: MONTE O CENÁRIO

Não há avalanche sem montanha. Chegou a hora de colocar em cena a encosta que Willie irá escalar. Precisaremos também colorir a superfície e completar a tela com um belo céu azul.

O título e os créditos foram exibidos. O jogador já leu as instruções e ouviu a execução do tema de abertura. Chegou a hora de abrir as cortinas — ou, no caso de *Avalanche*, a hora de desenhar o cenário. No Spectrum, este se movimenta da esquerda para a direita, enquanto a tela de instruções é arrastada em sentido oposto — ou seja, realiza-se um **SCROLL** horizontal. No MSX e no TRS-Color, a tela de instruções está no modo texto. Por isso, enquanto o cenário é colocado na tela gráfica, a página de instruções vai se apagando, cedendo-lhe lugar. O deslocamento do cenário para o vídeo se dá da mesma maneira.

O processo de montagem do cenário é bem simples. Um programa BASIC cria uma tabela com o contorno da montanha (para o TRS-Color isso foi dado no artigo anterior). A partir do conteúdo da tabela, a rotina em código desenha o perfil da encosta. O céu e a terra podem ser coloridos por meio de blocos gráficos, colocados acima e abaixo do perfil da encosta. À medida que o cenário é transferido para a tela, vamos retirando a página de instruções.



A rotina Assembly listada abaixo é responsável pela criação e pelo **SCROLL** horizontal do cenário.

```
 10 REM org 58303
 20 REM lsi ld a,16
 30 REM ld (57328),a
 40 REM ld ix,58034
 50 REM ld b,32
 60 REM mpi push bc
 70 REM call scl
 80 REM ld a,0
 90 REM ld (57329),a
100 REM ld a,(ix+0)
110 REM dec ix
120 REM cp 33
130 REM jr nz,lv
140 REM dec b
150 REM ld a,(57328)
160 REM dec a
170 REM ld (57328),a
180 REM ld a,1
190 REM ld (57329),a
200 REM lv ld a,(57328)
210 REM ld b,a
220 REM ld hl,31
230 REM ld a,45
240 REM call lg
250 REM ld bc,57264
260 REM ld a,(57329)
270 REM cp 1
280 REM jr nz,mp
290 REM ld bc,57272
300 REM mp ld a,44
310 REM call print
320 REM ld a,(57328)
330 REM ld b,a
340 REM ld a,23
350 REM sub b
360 REM ld b,a
370 REM ld a,32
380 REM ld de,32
390 REM add hl,de
400 REM call lg
410 REM pop bc
420 REM djnz mpi
430 REM ld hl,49
440 REM ld b,12
450 REM ld a,41
460 REM ld ix,57973
470 REM call me
480 REM ld hl,113
490 REM ld b,7
500 REM call me
510 REM call elb
520 REM ret
530 REM scl ld hl,16384
540 REM ld b,216
550 REM lpi ld c,31
560 REM lpj inc hl
570 REM ld a,(hl)
580 REM dec hl
590 REM ld(hl),a
600 REM inc hl
610 REM dec c
620 REM jr nz,lpj
630 REM inc hl
640 REM djnz lpi
650 REM ret
660 REM lg push bc
670 REM ld bc,15616
680 REM call print
690 REM ld de,32
700 REM add hl,de
710 REM pop bc
720 REM djnz lg
730 REM ret
740 REM elb ret
750 REM org 58146
760 REM me *
770 REM org 58217
780 REM print *
```

Este programa BASIC coloca na memória do microcomputador a tabela cujo conteúdo é responsável pela definição do perfil da encosta.



- COMO USAR UMA TABELA PARA DEFINIR O CONTORNO DA MONTANHA
- DESLOCAMENTO DO CENÁRIO
- O USO DOS BLOCOS GRÁFICOS
- COMO COLORIR OS ESPAÇOS
- REDEFINIÇÃO DE CARACTERES



```
 5 CLEAR 57000
10 FOR n=57973 TO 58034
20 READ a: POKE n,a: PRINT n;
" ";CHR$ a
30 NEXT n
40 DATA 83,67,79,82,69,45,48,
48,48,48,48,76,73,86,69,83,45
,53,71,65,77,69,32,79,86,69,
82,32,33,33,33,35,35,33,35,35
,35,33,35,35,33,35,33,35,35,
35,35,33,35,33,35,35,35,35,33
,35,35,33,35,35,35,54
```

Como de costume, o programa usa um laço **FOR...NEXT** para colocar dados na memória. O endereço inicial da tabela criada está na linha 10. A linha 5 protege o topo da memória para que os dados das outras tabelas não sejam apagados. Na linha **DATA** os valores 33 correspondem a porções inclinadas do perfil da encosta, e os números 35, a porções planas. Os demais valores são códigos das letras utilizadas para escrever o *score* e o número de vidas que restam a Willie.

A MONTAGEM DO CENÁRIO

Após a definição do endereço inicial, temos duas instruções cuja função é colocar no endereço 57328 o valor 16. Esse número corresponde à coordenada Y do extremo superior direito do horizonte. A posição 57328 será usada para armazenar números, como se fosse uma variável.

Em seguida, a instrução **ld ix,58034** coloca no par IX o último byte da tabela que define o perfil da encosta. Esse byte define o declive do extremo superior direito da montanha.

O registro B será usado como contador do número de colunas já desenhadas. Por isso, a instrução **ld b,32** coloca nele o valor 32, correspondente ao número de colunas da tela. A próxima linha guarda esse valor na pilha.

Em seguida, a sub-rotina scl é chamada. Ela se encarrega da translação do conteúdo da tela uma coluna para a esquerda — ou seja, realiza um **SCROLL** horizontal.

Uma segunda posição da memória, com endereço 57329, será usada como indicador. Ela informará à rotina se o

nível da encosta está diminuindo ou permanece plano naquela coluna. Se o byte for 0, a encosta é plana; se for 1, o nível está diminuindo. As instruções das linhas 80 e 90 colocam no endereço o valor inicial 0.

A instrução **ld a,(ix+0)** coloca no acumulador o último byte da tabela de contorno. Temos que usar o "+0" nessa instrução porque o endereçamento indireto com o registro IX deve ser indexado. Não existe uma instrução **ld a,(ix)**. A instrução **dec ix** subtrai uma unidade do conteúdo de IX, para que ele aponte para o próximo valor da tabela. Observe que esta será lida "de trás para a frente".

A instrução **cp 33** tem a função de comparar o conteudo do acumulador com 33, valor indicativo de que o nível da encosta está diminuindo. Se o acumulador não contém esse valor — ou seja, se a montanha é plana no local em questão —, a instrução **jr nz,lv** salta em direção ao rótulo **lv**.

Se o conteúdo do acumulador for 33, a instrução **jr nz,lv** é ignorada, pois o indicador de zeros foi ativado por **cp 33**. O contador B e a coordenada Y do perfil (armazenada em 57328) são diminuídos em uma unidade. O indicador armazenado no endereço 57329 passa a valer 1. Todo esse processo é executado pelas linhas 140 a 190.

A instrução **dec b** diminui o valor de B. As alterações do conteúdo dos endereços 57328 e 57329 são feitas com auxílio do acumulador, já que não existe uma instrução que, diretamente, diminua o valor de um byte da memória ou transfira um valor para determinado endereço. Não há instruções do tipo **dec 57328** ou **ld (57329),1**.

Quer a encosta da montanha apresente inclinação, quer continue plana, o programa continua na rotina **lv**.

A ROTINA lv

A linha 200 e a linha 210 colocam o valor da coordenada Y no registro B. Isso é feito com o auxílio do acumulador, já que o registro B não recebe o valor de um byte da memória via endereçamento indireto. Não há uma instrução **ld b,(57328)**.

A instrução **ld hl,31** coloca em HL a posição da memória de vídeo que corresponde ao canto superior direito da tela. **ld a,45** coloca no acumulador o valor 45, que, quando transferido para a tabela de atributos, resultará num caractere desenhado em ciano sobre fundo ciano. A rotina **lg** é chamada a seguir. Ela desenha uma coluna de blocos a partir da posição determinada por HL. A cor dos blocos é definida por A, enquanto o tamanho da coluna é estabelecido por B.

Quando o processador retorna da sub-rotina, a instrução **ld bc,57264** coloca em BC o endereço inicial do padrão do bloco usado para desenhar porções planas da encosta. O indicador guardado no endereço 57329 é colocado no acumulador e comparado ao número 1.

Se o valor do indicador for diferente de 1 — ou seja, se a montanha for plana neste local —, a instrução **jr nz,mp** faz com que o processador salte em direção ao rótulo **mp**. Se o valor do indicador for 1, o nível da encosta deve diminuir. A instrução **ld bc,(57329)** modifica o conteúdo de BC, de modo que esse par de registros passe a conter o endereço inicial do padrão do bloco usado para desenhar porções inclinadas da montanha. Os padrões dos blocos das porções planas e inclinadas da montanha são obtidos na tabela de padrões criada pelo programa listado no último artigo.

Quer a montanha seja plana ou inclinada neste ponto do desenho, o programa prossegue na rotina **mp**.

A ROTINA mp

Para desenhar os blocos que definem o perfil da encosta da montanha, determinando o limite entre o céu e a terra, o computador precisa de duas cores. Assim, a instrução **ld,a,44** atribui aos caracteres a cor verde. A cor do fundo permanece ciano. Em seguida, a sub-rotina **print**, listada no primeiro artigo da série *Avalanche*, é chamada para desenhar o bloco.

As instruções contidas nas linhas 320 e 330 colocam o valor da coordenada Y do perfil da encosta em B, com o auxílio do acumulador. Subtraindo esse valor de 23, obtém-se o número de blocos que devem ser desenhados abaixo do horizonte. O número 23, correspondente às linhas da tela do Spectrum, foi colocado em A para permitir a subtração realizada pelo comando **sub b**. Esse comando deixa o resultado da subtração no registro A. A instrução **ld b,a** é utilizada para trazer o resultado da subtração de volta para B — a rotina **lg** exige que o número de blocos a serem desenhados esteja em B. Note que todas as 24 linhas da tela do Spectrum são empregadas, incluindo as duas inferiores, normalmente reservadas para edição de linhas em BASIC.

A instrução **ld a,32** coloca em A o código de atributo correspondente a caractere de cor verde sobre fundo igualmente verde. Em seguida, o número 32 é colocado em DE e somado ao endereço de impressão na tela, que está em HL. Esse par de registros passa, então, a apontar para a próxima posição na vertical. A rotina **lg** é chamada a seguir para imprimir os blocos verdes abaixo do horizonte.

A instrução **pop bc** recupera da pilha o contador de colunas. A instrução **djnz** diminui seu valor em uma unidade, retornando ao início do programa para imprimir uma nova coluna, enquanto o contador não for zero.

O PLACAR

Precisamos reservar uma porção da tela para imprimir o *score* e o número de vidas que restam a Willie. Assim, na linha 430, 49 é colocado em HL, determinando a posição de impressão do placar. O número de caracteres impressos é colocado em B. A instrução **ld a,41** coloca em A o código de atributo correspondente a caractere azul sobre fundo ciano.

Depois, a instrução **ld ix,57973** coloca em IX o endereço do byte que será utilizado para armazenar o total de pontos obtidos pelo jogador. A rotina **me**, criada no primeiro artigo desta série, é, então, chamada. Ela traduz o *score* sob a forma de códigos ASCII, e o imprime na tela.

A instrução seguinte chama a sub-rotina **elb**, que cuida dos níveis de dificuldade do jogo, acrescentando os buracos e as cobras ao cenário. Essa rotina ainda não foi publicada e, como você pode observar, seu rótulo corresponde a um simples **ret**, no final da listagem. Por enquanto, o processador retornará imediatamente da rotina, sem nada executar. No momento apropriado, o comando **ret** será apagado pela verdadeira rotina **elb**.

Ao retornar da sub-rotina **elb**, o processador encontra uma nova instrução **ret**, que provoca um retorno ao interpretador BASIC. Quando o videogame estiver completo, a rotina de criação do cenário terá terminado e o processador retornará à rotina principal (responsável pelo controle do jogo), que chamará as sub-rotinas seguintes.

SCROLL HORIZONTAL

O rótulo **scl** marca o início da rotina que executa um deslocamento horizontal do conteúdo da tela para a direita. Esse processo — chamado geralmente

de **SCROLL** horizontal — permite que desenhemos o cenário, a partir da extremidade esquerda do vídeo, enquanto a página de instruções vai sendo retirada da tela.

A instrução **ld hl,16384** coloca o endereço inicial da memória de vídeo em HL. A instrução seguinte coloca em B o número de linhas da memória de vídeo e da tabela de atributos. O número de colunas por linha é colocado em C pela instrução **ld c,31**.

O apontador HL aumenta, então, em uma unidade, fazendo com que a instrução **ld a,(hl)** coloque no acumulador o conteúdo da segunda posição da tela. Em seguida, o conteúdo do par de registros HL é diminuído em uma unidade, voltando a apontar para a primeira posição da tela. A instrução **ld (hl),a** coloca ali o padrão que antes ocupava a posição seguinte.

O conteúdo do par HL é novamente aumentado, enquanto o conteúdo de C é diminuído. Se o registro C ainda não contém zero — ou seja, se o fim da linha não foi atingido —, a instrução **jr nz,lpj** retorna ao rótulo **lpj** para deslocar o conteúdo da coluna seguinte. Quando o fim da linha for alcançado, a instrução **jr nz,lpj** será ignorada e a instrução **inc hl** aumentará HL em uma unidade.

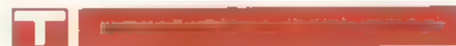
A instrução **djnz** sempre utiliza o par de registros BC. Como C é igual a zero quando se executa a linha 640 do programa, essa instrução diminui o conteúdo de B em uma unidade e volta ao rótulo **lpi** para deslocar uma nova linha. Depois que a última linha tiver sido deslocada para a direita, B conterá o valor 0, e a instrução **ret** provocará o retorno da sub-rotina.

A ROTINA lg

A primeira instrução dessa sub-rotina é **push bc**, que guarda o contador de colunas na pilha. A instrução **ld bc,15616** coloca em BC o endereço inicial da tabela da ROM em que estão contidos os padrões dos caracteres. O primeiro caractere ali representado é o espaço em branco. Assim, quando a linha 680 chamar a sub-rotina **print**, ela imprimirá um espaço de cor apropriada na tela. A instrução seguinte coloca 32 em DE. A instrução **add hl,de**, por sua vez, soma 32 ao conteúdo do par HL, de modo que ele aponte para o próximo bloco da coluna.

A instrução **pop bc** recupera o contador de colunas e a instrução **djnz** diminui seu valor em uma unidade, retornando ao rótudo **lg** a fim de imprimir mais um espaço em branco. O processo se repete até que o conteúdo de B seja reduzido a zero, indicando que o último bloco foi impresso.

O salto para **lg** não será executado e a instrução **ret** fará com que o processador retorne ao ponto de onde a sub-rotina foi chamada.



A rotina Assembly listada a seguir cria e desloca na tela do TRS-Color o cenário de *Avalanche*. Ela difere das demais devido às limitações do uso de cores nesse computador. Como a montanha é coberta de relva e cercada pelo mar azul, não temos alternativa senão pintar o céu de amarelo.

```
10   ORG 19109
20   JSR MODE
30   JSR GCLS
40   LDX #5631
50   LDY #17503
60   LDB #32
70   LOOP PSHS B
80   JSR SCROLL
90   JSR PRINT
100   PULS B
110   DECB
120   BNE LOOP
130   LDY #17604
140   LDX #1569
150   JSR PRSUN
160   RTS
170   MODE EQU 19182
180   GCLS EQU 19161
190   SCROLL EQU 19197
200   PRINT EQU 19218
```

Digite a rotina usando nosso programa Assembler. Grave o programa-fonte e depois monte-o. Grave também a rotina em código, usando o comando **CSAVEM**. A rotina não deve ser executada ainda, pois não funcionará sem os demais programas do artigo.

DESLOCAMENTO DO CENÁRIO

Após termos definido o endereço inicial, precisamos colocar o computador no modo gráfico e selecionar o conjunto de cores que vamos utilizar. Isso é feito pela sub-rotina **MODE**, chamada pelo comando **JSR MODE** na linha 20. A sub-rotina **GCLS** limpa a tela, colorindo-a de amarelo.

A instrução **LDX #5631** coloca em X o endereço do extremo superior esquerdo da tela. Essa área será ocupada pelo último byte do contorno da montanha antes de se iniciar o deslocamento do cenário. A instrução **LDY #17503** coloca em Y o endereço inicial da tabela do perfil da encosta. **LDB #32** coloca em B o número de colunas da tela. A instrução **PSHS B** guarda o contador de colunas na pilha da máquina, liberando o registro B.

A sub-rotina **SCROLL** é chamada para deslocar a coluna uma posição para a direita. Em seguida, a sub-rotina **PRINT** é chamada, imprimindo uma coluna de blocos verdes abaixo do horizonte. A instrução **PULS B**, na linha 100, recupera o contador de colunas da pilha da máquina. O contador é, então, diminuído em uma unidade por **DEC B**. A instrução **BNE LOOP** faz o processador voltar à linha 70 para imprimir uma nova coluna, até que o conteúdo de B acabe se tornando zero.

Quando isso ocorrer, o programa terá desenhado a última coluna, a instrução **BNE LOOP** será ignorada e o programa continuará na linha 140. Ali o registro Y recebe o endereço inicial do padrão do desenho do sol na tabela de blocos gráficos criada no artigo anterior. A linha seguinte coloca em X a posição do sol na tela gráfica. A instrução **JSR PRSUN** chama a sub-rotina que desenha o sol.

Como de costume, **RTS** provoca o retorno da sub-rotina ao interpretador BASIC — ou, quando o jogo estiver completo, ao programa principal.

Várias sub-rotinas chamadas não estão listadas no programa-fonte. Falsas instruções **EQV** informam seus endereços iniciais ao Assembler, para que ele possa calcular os saltos e desvios.

O INEVITÁVEL AMARELO

A rotina **GCLS** limpa a tela, colorindo-a totalmente de amarelo — cor do céu nesta versão do jogo.

```
10   ORG 19161
20   GCLS LDX #1536
30   LDA #85
40   GCLSI STA,X+
50   CMPX #7680
60   BLO GCLSI
70   RTS
```

A rotina começa colocando em X o endereço inicial da tela — 1536. A instrução **LDA #85** coloca no acumulador o código da cor amarela.

A instrução **STA ,X+** coloca o conteúdo do acumulador no endereço apontado por X e aumenta o valor de X em uma unidade. A instrução **CMPX #7680** compara X com 7680, primeiro byte acima do final da tela. A instrução **BLO GOLSI** retorna à linha 40 para colorir a próxima posição de tela, enquanto o final da memória de vídeo não tiver sido alcançado por X — ou seja, enquanto X

for menor que 7680. Quando isso acontece, a instrução **RTS** faz com que o processador volte ao ponto de onde a rotina foi chamada.

MODO GRÁFICO

As quatro sub rotinas seguintes podem ser montadas juntas, pois ocupam posições consecutivas na memória.

Para modificar o modo gráfico do TRS-Color temos que nos comunicar com dois circuitos integrados especiais que controlam o vídeo. Esses chips são o Gerador de Vídeo ou VDG (*Video Display Generator*) e o Multiplexador Síncrono de Endereços ou SAM (*Synchronous Address Multiplexor*).

Na linha 20, o valor 229 é colocado no acumulador. A seguir, a instrução **STA 65314** coloca o mesmo valor no endereço FF22. Essa posição da memória controla a saída de dados para o VDG e outros periféricos. Cada bit desse byte tem uma função diferente.

Na linha 30, o numero 229 — 11100101 — estabelece as linhas de comunicação. O "1" no bit sete determina o modo gráfico; "0" seria o modo de texto. Os bits seis e cinco, valendo "1", definem o modo gráfico P3. O bit três seleciona as cores — aqui, "0" dá verde, vermelho, amarelo e azul.

Os bits dois, um e zero não se referem ao VDG. Eles controlam o tamanho da memória RAM, a produção de sons e a saída para a impressora, respectivamente. Sua conformação usual é 101. Por isso, quando for modificar o conteúdo desse byte, coloque 101 nos três primeiros bits, a menos que haja alguma razão para não fazê-lo.

Quando alteramos as linhas de comunicação com o VDG, devemos fazer também algumas alterações nos bytes que se referem ao SAM. Esse chip tem um registro de dezesseis bits que corresponde às posições de memória que vão de FFCO a FFDF — intervalo que contém 32 bytes. Cada bit do registro existente nessas posições é ativado quando colocamos um valor nos bytes ímpares correspondentes, e apagado quando fazemos o mesmo nos bytes pares. O valor colocado nos bytes e no registro do VDG deve ser o mesmo. Os bits ativados pelas linhas 40 a 60 do programa-fonte informam ao SAM que a memória de vídeo começa no endereço 1536.

```
10   ORG 19182
20   MODE LDA #229
30   STA 65314
40   STA 65475
50   STA 65477
60   STA 65479
70   RTS
```

SCROLL HORIZONTAL

O rótulo **SCROLL** marca o início da rotina que executa um deslocamento horizontal do conteúdo da tela para a direita. Esse processo — chamado geralmente de **SCROLL** horizontal — permite que desenhemos o cenário a partir da extremidade esquerda do vídeo.

```
10   SCROLL PSHS X,Y
20   LDX #1536
30   LDY #1537
40   SCRO LDA ,Y+
50   STA ,X+
60   CMPX #7679
70   BLO SCRO
80   PULS Y,X
90   RTS
```

Essa rotina precisa utilizar os registros X e Y, mas números muito importantes foram guardados ali por outras



rotinas. Assim, a primeira providência do programa, na linha 10, é guardar esses valores na pilha da máquina, com a instrução **PSHS X,Y**.

As linhas 20 e 30 colocam em X e Y os endereços da primeira e da segunda posições da tela, respectivamente. A instrução **LDA ,Y+** coloca em A o conteúdo da posição apontada por Y, e Y aumenta em uma unidade.

A instrução **STA ,X+** coloca o conteúdo de A na posição apontada por X, e X aumenta em uma unidade. Portanto, na primeira passagem do processador, as linhas 40 e 50 colocam o conteúdo da segunda posição da tela dentro da primeira posição, atualizando ainda os valores dos dois apontadores.



A instrução **CMPX #7679** compara o conteúdo de X com o último endereço da memória de vídeo. Enquanto X contiver um valor inferior, a instrução **BLO SCROLL** envia o processador de volta à linha 10 para deslocar mais uma posição de memória para a esquerda.

A rotina não só desloca todo o conteúdo da tela uma posição para a esquerda como também coloca o que havia na última coluna da esquerda para a última coluna da direita, uma linha acima. Isso não tem importância, já que a última coluna da direita será apagada pelos blocos que desenham o cenário.

Quando o conteúdo da última posição da memória de vídeo for deslocado para a esquerda, o conteúdo original dos registros X e Y será recuperado da pilha pelo comando **PULS Y,X**. A instrução **RTS** da linha 90 provocará o retorno da sub-rotina.

O NOVO CENÁRIO

Os blocos gráficos que compõem o cenário preenchem a última coluna da direita, à medida que o conteúdo da tela vai sendo deslocado para a esquerda. A rotina responsável por esse processo é a seguinte.

```
10   PRINT PSHS X
20   LDA ,Y+
30   SUBA #33
40   BNE PRZ
50   PULS X
60   LEAX -256,X
70   PSHS Y
80   LDY #17536
90   LDB #8
100   PRI LDA ,Y+
110   STA ,X
120   LEAX 32,X
130   DECB
140   BNE PRI
150   PULS Y
160   PRZ CLR ,X
170   LEAX 32,X
180   CMPX #7680
190   BLC PRZ
200   PULS X
```

Desta vez a rotina usará o valor contido no apontador Y. Se estivermos desenhando uma porção inclinada da montanha, poderá ser necessário modificar a altura do horizonte contida no registro X. Mas, por enquanto, a instrução **PSH X** guarda o conteúdo de X na pilha da máquina.

A instrução **LDA ,Y+** coloca em A o conteúdo do endereço indicado por Y e o apontador aumenta em uma unidade. A seguir, a instrução **SUBA #33** subtrai 33 unidades de A.

O número 33 é o código ASCII do ponto de exclamação e, na tabela criada pelo programa do artigo anterior, significa que a encosta é inclinada naquela posição. Se este não for o caso, a subtração não resulta em zero, o que faz a instrução **BNE PRZ** provocar um salto para a linha 160. Se A contiver o valor 33, indicando inclinação da encosta, a instrução **BNE** é ignorada e o programa continua.

A instrução **PULS X** recupera da pilha a altura do horizonte, e a instrução seguinte subtrai 256 unidades de X. O número 256 é igual a 8 × 32 o apontador X subiu, assim, oito linhas (ou um bloco) na tela. O novo valor de X é recolocado na pilha da máquina para ser usado no desenho da coluna seguinte. O valor do apontador Y que aponta para a tabela de contorno da montanha — também é guardado na pilha.

A instrução **LDY #17536** coloca em Y o endereço inicial do padrão do bloco de uma porção inclinada da montanha. O bloco é desenhado na posição apontada por X. Neste modo gráfico os pontos são criados em grupos de dois, com base no valor de dois bits. Dois bits podem ter quatro valores diferentes, um para cada cor. A organização da memória de vídeo foi explicada no artigo da página 86.

A instrução **DECB** diminui o conteúdo do contador B em uma unidade e, enquanto esse contador não tiver sido reduzido a zero, a instrução **BNE PRI** fará com que o processador volte à linha 100 para desenhar a próxima linha. Quando B contiver zero, a última linha terá sido desenhada.

Em seguida, a instrução **PULS Y** recupera da pilha o apontador da tabela de contorno da montanha. O programa continua na linha rotulada **PRZ**. Esta é a rotina para a qual o programa saltaria se a montanha fosse plana neste ponto (veja linha 40). Sua função é desenhar uma coluna de blocos verdes, recobrindo parte da montanha.

COMO FUNCIONA

A instrução **CLR ,X** limpa o conteúdo do endereço apontado por X. Limpar — ou seja, tornar igual a zero — o conteúdo de uma posição de memória equivale, no caso, a pintar aquela posição com a cor verde. A instrução **LEAX 32,X** soma 32 ao apontador X, fazendo-o descer uma linha na tela.

A instrução **CMPX #7680** verifica se o apontador ultrapassou o final da memória de vídeo. Enquanto isso não ocorrer, a instrução **BLO PRZ** fará o processador voltar ao rótulo **PRZ** para desenhar mais uma linha de oito pontinhos verdes. Ao se completar a coluna de blocos verdes, a instrução **PULS X** recupera da pilha o valor da altura do horizonte. A instrução **RTS** marca o final da sub-rotina.

O SOL

Os dados necessários ao desenho do sol (padrões e posições na tela) se encontram na tabela criada no artigo anterior. Esses dados são utilizados pela rotina Assembly listada a seguir, que preenche um espaço de 32 por 30 pontos no céu.

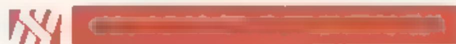
```
10  PRSUN LDB #30
20  PRSUNI PSHS B
30  LDB #4
40  PRSUNZ LDA ,Y+
50  STA ,X+
60  DECB
70  BNE PRSUNZ
80  LEAX 28,X
90  PULCS B
100  DECB
110  BNE PRSUNI
120  RTS
```

A instrução **LDB #30** coloca no contador B o número de linhas que serão preenchidas para desenhar o sol. Depois, a instrução **PULS B** guarda o contador na pilha da máquina. A linha seguinte coloca o número 4 no registro B, definindo em quatro bytes — ou 4 × 8 = 32 bits — a largura do desenho. Mesmo que quiséssemos desenhar um sol com 30 × 30 pontos, precisaríamos de um quadriculado de 32 × 30 pontos — usaríamos, no caso, quatro bytes, deixando duas colunas com a cor de fundo.

Como de costume, a instrução **LDA ,Y+** obtém o padrão de uma parte dos desenhos no endereço apontado por Y, colocando-o em A e aumentando em uma unidade o apontador. A instrução **STA ,X+** coloca esse padrão na tela, na posição apontada por X, aumentando também em uma unidade o valor desse apontador. A instrução **DECB** diminui o valor do contador e, enquanto B não for zero, a instrução **BNE PRSUNZ** retorna à linha 40, fazendo com que quatro bytes sejam colocados na tela.

Quando o último byte da linha tiver sido desenhado, a instrução **LEAX 28,X** soma 28 ao conteúdo de X, de modo que ele aponte para o início da próxima linha. O contador de linhas é recuperado da pilha por **PULS B** e diminuído em uma unidade por **DECB**. Enquanto as trinta colunas não tiverem sido desenhadas, a instrução **BNE PRSUNI** faz com que o processador volte à linha 20, repetindo o processo.

Se a última linha foi traçada, B reduz-se a zero, a instrução da linha 110 é ignorada e o programa encontra a instrução **RTS**, que provoca o retorno da sub-rotina.



Quando escrevemos a página de instruções, a tela estava no modo texto de quarenta colunas, totalmente inadequado para o cenário de um jogo de ação. A rotina Assembly a seguir seleciona o modo gráfico de alta resolução.

```
10   org -12144
20   ld hl,62441
30   ld (hl),4
40   inc hl
50   ld (hl),7
60   inc hl
70   ld (hl),7
80   call 114
90   ld a,226
100   ld (62432),a
110   call 105
120   ret
130   end
```

Para selecionar as cores da tela, é preciso modificar o conteúdo de algumas posições da RAM, nas quais o sistema armazena valores usados no controle das diversas operações do micro.

A instrução que se segue à definição do endereço inicial coloca em HL o endereço do byte que determina a cor de frente usada na tela. Depois, a instrução **ld (hl),4** utiliza o endereçamento indireto para colocar ali o código da cor azul-escuro.

O par HL, usado como apontador, tem seu conteúdo aumentado em uma unidade pela instrução **inc hl**, passando a apontar para o próximo endereço, que determina a cor de fundo da tela. Esse byte recebe o valor 7, que é o código da cor ciano. Duas outras instruções colocam o mesmo código na posição seguinte, que determina qual será a cor da moldura da tela.

A instrução **call 114** é responsável pela modificação das cores e do modo de exibição da tela. Ela chama uma sub-rotina do sistema, que coloca o computador no modo gráfico de alta resolução. Essa rotina equivale a um comando **SCREEN 2,0**, ou seja, ela não somente coloca o computador no modo gráfico como também limpa a tela, preenche as tabelas da VRAM com seu conteúdo padrão e, ainda, acerta os conteúdos do VDP (*Video Display Processor*), chip de imagem do MSX.

A rotina chamada na linha 80, contudo, preparou o VDP para exibir sprites pequenos (8 × 8 pontos), quando nós precisamos de sprites grandes (16 × 16 pontos). Para fazer os acertos necessários, teremos que modificar o conteúdo de um dos registros do VDP. Estes podem ser alterados através das posições de memória que vão de 62431 a 62438 — o que corresponde a oito registros de oito bits. As funções de cada registro do VDP serão comentadas em outro artigo. Por hora, interessa-nos apenas o registro 1, que modificaremos através do endereço 62432.

As instruções das linhas 90 e 100 da listagem colocam nesse registro o valor 226. Note que o uso do acumulador é temporário, justificando-se pela inexistência de uma instrução que coloque um número diretamente em uma posição de memória. Não há uma instrução **ld (62432),226**.

Cada bit do registo 1 do VDP tem uma função. O bit 7 indica o tamanho da VRAM utilizada, o bit 6 liga e desliga a tela, o bit 5 seleciona o modo de interrupção, os bits 3 e 4 determinam o tipo de tela (dois bits nos dão quatro opções: telas 0, 1, 2 e 3), o bit 2 não é utilizado, o bit 1 controla o tamanho lógico do sprite (8 × 8 ou 16 × 16 pontos) e o bit 0, finalmente, controla o tamanho físico do sprite (normal ou ampliado). Assim, como 226 é 11110010 em sistema binário, quando colocado no registro 1, ele determina uma VRAM de dezesseis Kbytes, tela ligada, com possibilidade de interrupção, modo gráfico de alta resolução e sprites grandes não ampliados.

A sub-rotina chamada na linha 80 preencheu a tabela de atributos de sprites (veja o artigo da página 808) com nomes compatíveis com sprites pequenos. A instrução **call 105** chama, então, outra sub-rotina da ROM, que preenche a mesma tabela com valores compatíveis com sprites de 16 × 16 pontos. Na realidade, essa sub-rotina preenche a tabela de atributos com seu conteúdo padrão, conforme os valores dos registros do VDP. Depois disso, nossa rotina termina com a instrução **ret**.

TABELA DE PADRÕES

A tela de alta resolução dos microcomputadores da linha MSX é organizada em cinco tabelas: nomes, padrões, cores, atributos de sprites e padrões de sprites. Os efeitos visuais do videogame são criados por intermédio da modifi-

cação dos valores contidos nessas tabelas.

A rotina listada a seguir transfere para a tabela de padrões os blocos gráficos criados pelo programa BASIC do artigo anterior. Como os mesmos blocos são transferidos para a tabela de padrões de sprites, podemos utilizar as figuras que criamos tanto na forma de blocos gráficos como na forma de sprites. Observe que o programa exige que o banco de blocos anteriormente criado esteja na memória.

```
10    org -12121
20    ld de,(62411)
30    ld hl,-15100
40    ld bc,640
50    push bc
60    push hl
70    push de
80    call 92
90    pop de
100    ld hl,2048
110    add hl,de
120    ld d,h
130    ld e,l
140    pop hl
150    pop bc
160    push bc
170    push hl
180    push de
190    call 92
200    pop de
210    ld hl,2048
220    add hl,de
230    ld d,h
240    ld e,l
250    pop hl
260    pop bc
270    push bc
280    push hl
290    call 92
300    ld de,(62415)
310    pop hl
320    pop bc
330    call 92
340    ret
350    end
```

Para transferir os padrões da RAM para a VRAM, utilizamos a sub-rotina da ROM que fica no endereço 92. Usamos a mesma sub-rotina para criar a página inicial e escrever as instruções.

A instrução **ld de,(62411)** coloca em DE o endereço inicial da tabela de padrões. Esse endereço fica armazenado nos bytes 62411 e 62412 e equivale a **BASE(12)**. Note que a instrução transfere dezesseis bits, ou seja, um par de bytes para um par de registros. O registro E recebe o conteudo de 62411, byte menos significativo. D recebe o byte mais significativo, 62412.

A instrução seguinte coloca em HL o endereço inicial do banco de blocos que criamos na memória. O número de bytes a serem transferidos para a VRAM é colocado em BC por **ld bc,700**.

A sub-rotina que começa no endereço 92 baseia-se no conteúdo dos registros DE, HL e BC para efetuar a transferência. DE deve conter o endereço inicial da porção da VRAM a ser modificada; HL, o endereço inicial da porção da RAM que vai ser transferida; e BC, o número de bytes. Como a sub-rotina altera o conteúdo desses mesmos registros no decorrer de seu funcionamento, eles são guardados na pilha, para eventual uso futuro. As instruções **push bc, push hl** e **push de** tratam disso, antes que **call 92** mande o processador executar a sub-rotina.



A tela de nomes da tela de alta resolução é dividida em três porções, cada uma representando os padrões contidos no terço correspondente da tabela de padrões. Como usaremos toda a tela, precisaremos ter três cópias do banco de blocos na tabela de padrões.

Para fazer a segunda cópia, o endereço inicial da tabela de padrões é recuperado como **pop de**. O par HL recebe, então, o valor 2048, que corresponde à posição inicial do segundo terço da tabela de padrões. Esse valor é somado ao endereço inicial da tabela, para obtermos o endereço inicial do terço médio na VRAM. A instrução que faz esta soma — **add hl,de** — deixa o resultado da operação em HL.

Precisamos do endereço inicial em DE. Como não existem as instruções **add de,hl** e **ld de,hl**, para transferir o resultado para o par HL utilizamos duas instruções: **ld d,h** e **ld e,1**.

O endereço inicial do banco de blocos na RAM é recuperado da pilha com **pop hl** e o número de bytes a serem transferidos para a VRAM, como **pop bc**. Observe que a ordem de recuperação dos pares de registro da pilha é a mesma de sua colocação. Os registros são guardados na pilha novamente, antes que **call 92** volte a transferir o banco de blocos para a VRAM.

Obtém-se a terceira cópia do banco de blocos de maneira muito parecida. O endereço inicial do terço médio é recuperado e somado a 2048, resultando no endereço correspondente ao terço inferior. HL e BC são recuperados da pilha e a rotina da memória ROM é executada novamente. Desta vez, somente BC e HL são guardados na pilha.

Uma quarta cópia do banco de blocos deve ser feita na tabela de padrões de sprites. O endereço inicial da tabela é armazenado nos endereços 62415 e 62416. A instrução **ld de,(62415)** transfere esses dois bytes para DE, como foi explicado anteriormente.

As instruções **pop hl** e **pop bc** recuperam da pilha o endereço inicial e o tamanho do banco. A cópia é feita com **call 92**. A rotina termina com uma instrução **ret**.

A TABELA DE CORES

Se não colocarmos valores adequados na tabela de cores, nenhum bloco gráfico será mostrado no vídeo. Um novo programa BASIC, que coloca os códigos de cores no topo da memória, está listado no final do artigo. Os valores são transferidos para a tabela de cores da tela gráfica por esta rotina:

```
org -12066
ld de,(62409)
ld hl,-16100
ld bc,672
push bc
push hl
push de
call 92
pop de
 ld hl,2048
 add hl,de
 ld d,h
 ld e,l
 pop hl
 pop bc
 push bc
 push hl
 push de
 call 92
 pop de
 ld hl,2048
 add hl,de
 ld d,h
 ld e,l
 pop hl
 pop bc
 call 92
 ret
 end
```

Essa rotina é bem parecida com a anterior. Ela começa colocando em DE o endereço inicial da tabela de cores, armazenado nos endereços 62409 e 62410. HL recebe o endereço inicial da lista de códigos de cores que será criada pelo programa BASIC no final do artigo. BC recebe o número de bytes a serem transferidos. Os conteúdos desses três pares de registros são guardados na pilha e a rotina do endereço 92 é chamada novamente.

Utiliza-se o mesmo processo explicado anteriormente para a obtenção de três cópias do banco de cores — cada uma correspondendo a um terço do vídeo. Sprites serão coloridos no momento em que surgirem no vídeo.

SCROLL HORIZONTAL

Para realizar o deslocamento da tela para a direita, usaremos um processo muito semelhante ao que foi descrito no artigo da página 213.

As três rotinas listadas abaixo realizam o processo da translação do conteúdo da tela gráfica:

```
10    org -12022
20    ld hl,(62407)
30    ld de,-6000
40    ld bc,768
50    call 89
60    ret
70    org -12009
80    ld de,-5233
90    ld hl,-5234
100    ld b,24
110    loop push bc
120    ld a,(de)
130    ld bc,31
140    lddr
150    ld (de),a
160    dec hl
170    dec de
180    pop bc
190    djnz loop
200    ret
210    org -11987
220    ld de,(62407)
230    ld hl,-6000
240    ld bc,768
250    call 92
260    ret
270    end
```

A primeira rotina transfere o conteúdo da tabela de nomes da tela de alta resolução para um *buffer* localizado no topo da memória.

O endereço inicial dessa tabela na VRAM fica guardado nos bytes 62407 e 62408 da RAM. A instrução da linha 20 do programa coloca esse valor no par de registros HL. O endereço inicial do buffer é colocado em DE por **ld de,-6000**. O número de bytes a serem transferidos vai para o par de registros BC. A instrução **call 89** chama, então, a sub-rotina da ROM que transfere numeros da VRAM para a RAM.

A segunda sub-rotina, que começa na linha 70, promove o deslocamento do conteúdo do buffer para a direita.

Os endereços dos dois últimos bytes do buffer são colocados em DE e HL. O número de linhas da tela vai para B. Como BC será utilizado adiante, a instrução **push bc** armazena o contador de linhas na pilha.

O conteúdo da última posição da tela e guardado no acumulador por **ld a,(de)**. O par BC recebe o número de colunas de uma linha — 31. A instruçao **lddr** coloca no endereço apontado por DE o conteúdo do endereço apontado por HL e subtrai uma unidade de cada um dos apontadores e do contador BC. O processo e repetido automaticamente até que BC seja zero. A instrução da linha 140 desloca, assim, toda uma linha de blocos para a direita.

O conteúdo da última posição da linha é colocado na primeira posição, agora apontada por DE. As instruções **dec hl e dec de** fazem com que esses registros apontem para as duas últimas posições da próxima linha a ser deslocada. No nosso caso, ela fica imediatamente acima da primeira.

O contador de linhas é recuperado da pilha e **djnz** repete o laço **loop** até que todas as 24 linhas de blocos gráficos que compõem o buffer tenham sido deslocadas.

A última rotina, que começa na linha 210, transfere o conteúdo do buffer de volta para a tabela de nomes. Suas instruções são parecidas com as da rotina da linha 10, só que DE recebe o endereço da tabela na VRAM e HL recebe o endereço do buffer. A rotina chamada fica no endereço 92 da ROM.

A ROTINA PRINCIPAL

Para desenhar a montanha que Willie deve escalar, o programa usa a seguinte rotina:

```
org 53563
ld a,255
ld hl,(62407)
ld bc,768
call 86
call -12121
call -12066
call 12022
ld a,8
 ld ( 5230),a
 ld a,0
 ld (-5229),a
 ld hl,-6001
 ld b,33
 mp1 push bc
 push hl
 call -11987
 call 12009
 pop hl
 ld a,(-5229)
 cp .
 jr nz,lg
 ld a,(-5230)
 inc a
 ld ( 5230),a
 lg ld a,0
 ld (-5229),a
 ld a,(hl)
 dec hl
 push hl
 cp 33
 jr nz,lv
 ld a,1
 ld ( 5229),a
 lv ld a,(-5230)
 ld b,a
 ld hl,-6000
 ld a,255
 call lg
 ld b,48
 ld a,( 5229)
 cp 1
 jr nz,no
```

```
440    ld b,52
450    no ld a,b
460    ld (hl),a
470    ld a,(-5230)
480    ld b,a
490    ld a,23
500    sub b
510    ld b,a
520    ld a,81
530    ld de,32
540    add hl,de
550    call lg
560    pop hl
570    pop bc
580    djnz mpi
590    ret
600    lg ld (hl),a
610    ld de,32
620    add hl,de
630    djnz lg
640    ret
650    end
```

A tabela de nomes será utilizada para representar blocos gráficos. Estes devem estar devidamente representados nas tabelas de padrões e de cores. Para entender o funcionamento das rotinas gráficas é necessário conhecer a organização da VRAM.

A primeira coisa que a rotina principal faz é preencher a tabela de nomes com o número 255. Isso equivale a encher a tela com o bloco gráfico de código 255, que, no nosso caso, é um bloco ciano. O acumulador recebe o código do bloco, 255. A instrução **ld hl,(62407)** coloca o endereço inicial da tabela de nomes em HL. BC recebe o número de bytes da tabela = 768

A rotina da memória ROM chamada por **call 86** preenche com o conteúdo de A uma porção da VRAM. O par de registros HL aponta o endereço inicial da porção e o par BC, o número de bytes que sofrerão alterações

A linha 60 chama a sub-rotina que gera os padrões dos bytes, e a 70, a sub-rotina que cuida da tabela de cores. A sub-rotina do endereço − 12022 coloca o conteúdo da tabela de nomes no buffer de deslocamento.

As linhas 90 e 100 colocam no endereço -5230 o número da linha em que colocaremos o horizonte. O endereço -5229 recebe o valor zero. Esse byte servirá de indicador de inclinação da montanha. Se valer zero, ela será plana, se valer um, será inclinada.

O par HL recebe o último valor da tabela de perfil da encosta. Essa tabela será gerada por um programa BASIC listado no final do artigo. O perfil é definido por meio de códigos que indicam se a montanha é plana ou inclinada num determinado local.

Na linha 140, B recebe o número de colunas, que é logo colocado na pilha para liberar o uso desse registro. As linhas 170 e 180 chamam as rotinas que copiam o buffer na tabela de nomes e realizam o deslocamento

As linhas 200 a 250 verificam se o indicador de inclinação, no endereço -5229, foi modificado anteriormente. Quando isso acontecer, a linha do horizonte deve ser alterada no endereço -5230, o que é feito pelas linhas 230 a 250. A seguir, o indicador recebe novamente o valor zero.

A instrução **ld a,(hl)** coloca em A o valor da tabela de perfil da encosta apontado por HL. Em seguida, esse par de registros é diminuído em uma unidade e guardado na pilha.

O conteúdo de A é comparado a 33 — valor que indica região inclinada. Se A tiver outro valor, a instrução **jr nz,lv** salta para o rótulo **lv**. Caso contrário, as linhas 330 e 340 colocam o número 1 no indicador de inclinação.

A ROTINA lv

As instruções das linhas 350 e 360 colocam em B o número da linha do horizonte. Isso é feito com o auxílio do acumulador, já que não existe uma instrução que coloque o conteúdo de um endereço diretamente em B. HL recebe o endereço inicial do buffer, que corresponde ao topo da primeira coluna da tela. A instrução **ld a,255** coloca o código do bloco ciano no acumulador. Esse bloco será usado para desenhar o céu. A rotina **lg** desenha uma coluna de blocos ciano.

A instrução **ld b,48** coloca em B o código do bloco que representa uma porção plana da montanha. As duas linhas seguintes verificam o conteúdo do indicador de inclinação. Se a encosta for plana nessa coluna, a instrução **jr nz,no** desvia o programa para o rótulo **no** Se for inclinada, B recebe o código do bloco adequado e a rotina passa ao rotulo **no**.

A ROTINA no

As linhas 450 e 460 colocam o bloco adequado na posição apontada pelo par de registros HL. As duas linhas que se seguem colocam em B o número da linha do horizonte. O registro A recebe o número de linhas da tela e a instrução da linha 500 subtrai o numero da linha de horizonte desse valor. Calcula-se, assim, o número de blocos que devem ser colocados abaixo da linha de horizonte nesta coluna

O resultado da operação fica em A; a instrução **ld b,a** transfere-o para B. A instrução **ld a,81** coloca em A o codigo do caractere verde, que preenche a encosta. O par de registros DE recebe o número 32 e **add hl,de** soma esse valor ao conteúdo de HL, para que este aponte para a linha seguinte. A rotina **lg** é chamada novamente para desenhar uma coluna de blocos.

Para finalizar, a posição do apontador da tabela de perfil da encosta da montanha é recuperado da pilha por **pop hl**. O contador de colunas também sai da pilha com **pop bc**. A instrução **djnz** repete o processo até que as 32 colunas do desenho tenham sido traçadas e deslocadas na tela.

A ROTINA lg

Essa rotina desenha uma coluna de blocos. O código do bloco deve estar no acumulador; o endereço inicial da coluna no buffer deve estar em HL; o número de linhas, em B.

A linha 600 coloca o código do bloco gráfico no endereço dado por HL. DE recebe o número 32, que é logo somado ao conteúdo de HL. A instrução **djnz** repete o processo de acordo com o conteúdo de B.

AS TABELAS

Este programa cria a tabela de cores na RAM:

```
5 CLEAR 200, 16100
10 FOR I=0 TO 20
20 READ A
30 FOR J=0 TO 31
40 POKE  16100+I*32+J,A
50 NEXT J,I
100 DATA 23,23,23,103,103,247,2
47,23,23,199,199,199,55,55,167,
135,215,151,244,244,51
```

A tabela criada tem 672 bytes. Cada valor na linha **DATA** 100 corresponde a um código de cor, que é repetido 32 vezes, uma para cada linha do bloco gráfico.

O programa a seguir cria a tabela do perfil da encosta:

```
10 FOR I=0 TO 31
20 READ A:POKE  6032+I,A
30 NEXT
100 DATA 33,33,35,35,33,35,35,3
5,33,35,35,33,35,33,35,35,35,35
,33,35,33,35,35,35,35,33,35,35,
33,35,35,35
```

Cada valor da linha **DATA** 100 corresponde a uma coluna do desenho. Um número 33 significa porção inclinada, e um 35, porção plana.

UMA AGENDA ELETRÔNICA (1)

Com nosso programa para calendário e agenda, você poderá planejar seus compromissos diários e manter um registro de datas especiais. Aproveite a oportunidade e organize-se!



Você é daquele tipo de pessoa que sempre se esquece do aniversário da esposa ou só se lembra da hora marcada no dentista quando já é tarde demais? Ou, ainda, que se assusta ao descobrir que uma conta de luz está vencida há mais de um mês?

O programa que apresentamos neste artigo cuidará de todos os seus compromissos. Com ele, você não terá mais desculpas para faltar a encontros ou deixar de pagar suas contas na data certa. Tendo uma impressora, você poderá, inclusive, fazer uma cópia da agenda em papel, para consultá-la sempre que estiver longe do computador.

CALENDÁRIO AUTOMÁTICO

Este programa coloca à sua disposição um calendário ou uma agenda. A primeira opção é a mais simples: mostra um calendário para qualquer mês dos anos de 1753 a 29999. A apresentação do calendário é a usual, com os dias da semana no topo e os dias do mês abaixo. O programa leva em consideração, automaticamente, os anos bissextos, e indica a data do domingo de Páscoa no mês correspondente.

É possível, ainda, imprimir um calendário para o ano todo — alternativa especialmente útil se você pretende colocá-lo em sua escrivaninha ou pendurá-lo na parede.

A AGENDA ELETRÔNICA

Nossa agenda permitirá que você organize melhor seu dia-a-dia, mantendo um registro de todos os seus compromissos. As entradas são feitas sob quatro títulos — Finanças, Encontros, Celebrações e Feriados —, o que torna mais fácil encontrar um determinado tipo de informação.

É muito simples dar entrada à informação e, como vantagem adicional, o programa requer um único registro dos eventos regulares. Assim, aniversários e contas a pagar podem ser anotados uma só vez, no dia correto, em todos os meses ou anos subseqüentes. Quando se digita um dado de Finanças, por exemplo, o programa pergunta se o evento é único (apenas para aquela data específica), mensal, trimestral ou anual. Se você está entrando dados sobre um pagamento mensal, apenas digite **M** para mensal, em seguida o nome **ALUGUEL** e, finalmente, o dia do vencimento. A palavra **ALUGUEL** aparecerá naquele dia em todos os meses seguintes.

A opção Celebrações inclui datas como aniversários de nascimento, casamento, enfim, todo evento que tenha

ocorrência anual. Os Encontros e Feriados são tratados como eventos únicos.

Grave os dados logo depois de entrá-los, usando a opção **GRAVAR**, para evitar que alguma informação se perca. Eles são armazenados em um arquivo à parte, chamado DIÁRIO, e podem ser carregados, corrigidos ou apagados a qualquer momento, o que torna muito fácil a atualização da agenda.

O USO DO PROGRAMA

Depois de digitar alguns dados, você poderá observar o funcionamento da agenda. Se teclar o número do mês e ano, verá todas as entradas para aquele mês, incluindo compromissos financeiros ou celebrações levadas de um mês a outro pelo programa.

As entradas são agrupadas nas diferentes categorias. Antes de exibir determinado grupo na tela, o programa espera que uma tecla seja pressionada. Dessa maneira, nenhuma entrada será retirada sem que você possa vê-la. Se dispuser de uma impressora, copie todas as entradas em papel, sempre que assim julgar necessário.

Se você decidir voltar à opção de calendário e apontar o mesmo mês, poderá destacar as datas correspondentes a compromissos na agenda pressionando, para cada uma das categorias, as teclas $, E, C ou F.

A escolha de uma ou de outra opção do programa depende daquilo que se quer encontrar. Se você pretende examinar a programação para um determinado mês, a escolha mais adequada será a da agenda. Mas você pode precisar exclusivamente de informações sobre os eventos financeiros do ano, para fazer um planejamento. Nesse caso, será melhor selecionar o calendário para um dos meses, destacar os compromissos comerciais teclando $, passar para o mês seguinte e repetir a operação até completar o período desejado.

Como você verá, nosso programa é muito versátil. Você descobrirá várias outras possibilidades tão logo comece a usá-lo.

PRIMEIRA PARTE

Dividimos o programa em três partes. A primeira, que apresentamos a seguir, contém várias rotinas e a tela do menu principal. As outras duas partes, incluindo instruções mais detalhadas de como usar o programa, serão dadas em artigos futuros.

Não se esqueça de que deve gravar esta parte do programa para acrescentá-la às próximas.



```
 10 DATA 2,4,1,3,7,31,28,31,30
 ,31,30,31,31,30,31,30,31
 20 DATA "MENS","TRIM","ANUA",
 "UNIC"
 30 BORDER 0: PAPER 0: INK 7:
 CLS
 40 CLS
 50 CLEAR : LET P=2
 60 LET Z$="
                 "
 70 DIM O$(1,31): DIM Q(4):
 DIM L$(4,150,31): DIM T$(4.12
 ): DIM C(4): DIM Z(5)
 80 FOR n=1 TO 5: READ Z(n):
 NEXT n
 90 DIM D(12): FOR n=1 TO 12:
 READ D(n)· NEXT n
100 LET M$="Janeiro Fevereiro
Marco    Abril    Maio      Jun
ho    Julho    Agosto   Setemb
ro Outubro  Novembro Dezembro
"
110 LET m$="DomSegTerQuaQuiSex
Sab"
120 DIM P$(4,4): FOR n=1 TO 4:
READ P$(n): NEXT n
130 LET T$(1)="Financas": LET
T$(2)="Apontamentos": LET T$(3
)="Celebracoes": LET T$(4)="Fe
riados"
140 DEF FN M(A)=((A/K2-INT (A/
K2))*K2)
150 LET SV=0: LET MO=0: LET DA
=0
160 PRINT "HA ALGUMA LISTA DE
DADOS GRAVADA(S/N)  ?"· LET K$
="sn"
170 CLS : GOSUB 990  CLS   LET
P=2
180 IF C=1 THEN  GOSUB 760
190 IF C=2 THEN  GOSUB 1760
200 IF C=3 THEN  GOSUB 2240
210 IF C>3 AND C<8 THEN  LET K
B=C-3: GOSUB 1140· LET SV=1
220 IF C=8 THEN  GOSUB 1610:
LET SV=0
230 IF C=9 AND SV=1 THEN
PRINT : PRINT "VOCE NAO GRAVOU
 AS ALTERACOES"'"CONFIRMA A SA
IDA ?": LET K$="sn": GOSUB
1480: IF KB=2 THEN  LET C=0
240 IF C<>9 THEN  GOTO 170
250 CLS : PRINT "ADEUS !"
260 STOP
270 LET MX=0: LET A2=0
280 LET K2=4: IF FN M(YR)=0
THEN  LET A2=1
290 LET K2=100: IF FN M(YR)=0
THEN  LET A2=0
```

```
300 LET K2=400: IF FN M(YR)=0 THEN  LET A2=1
310 IF KB=2 THEN  LET MX=A2+28
320 IF KB=0 THEN  LET KB=1
330 IF KB<>2 THEN  LET MX=D(KB)
340 LET KB=MX: RETURN
350 LET RS=0
360 IF DA=DE AND MO=ME AND KB=5 THEN  LET KB=5: RETURN
370 IF KB=5 THEN  LET KB=7: RETURN
380 LET RS=Z(KB)
390 IF Q(KB)=0 THEN  GOTO 450
400 LET M4=Q(KB)
410 FOR I=1 TO M4
420 LET K$=L$(KB,I): LET K2=3: GOSUB 470: IF VAL K$(2 TO 3)<>DA THEN  LET K2=0
430 IF K2=1 THEN  LET KB=7: RETURN
440 NEXT I
450 LET KB=RS
460 RETURN
470 IF KB<>1 THEN  GOTO 520
480 IF VAL K$(1)=3 THEN  GOTO 540
490 IF VAL K$(1)=4 THEN  GOTO 530
500 IF VAL K$(1)=1 AND VAL K$(2 TO 3)=DA THEN  LET K2=1: RETURN
510 IF VAL K$(1)=2 AND FN M((((YR-VAL K$(6 TO 9))*12)+(12-VAL K$(4 TO 5))+MO))=0 THEN LET K2=1: RETURN
520 IF KB=3 THEN  GOTO 540
530 IF VAL K$(2 TO 3)=DA AND VAL K$(4 TO 5)=MO AND VAL K$(6 TO 9)=YR THEN  LET K2=1: RETURN
540 IF VAL K$(4 TO 5)=MO THEN LET K2=1: RETURN
550 LET K2=0: RETURN
560 LET Y2=0: LET D2=0: LET M2=0
570 LET Y2=YR-1
580 LET D2=Y2*365+INT (Y2/4)-INT (Y2/100)+INT (Y2/400)
590 IF MO=1 THEN  GOTO 630
600 FOR m=1 TO MO-1
610 LET KB=m: GOSUB 270: LET D2=D2+KB
620 NEXT m
630 LET KB=D2+DA: RETURN
640 LET MS=MO: LET DS=DA
650 LET DA=1: LET MO=3: GOSUB 560: LET K2=7: LET DE=FN M(KB)
660 LET N2=(INT (YR/100))-16: LET C2=3+N2-INT ((N2+1)/3)-INT (N2/4)
670 LET K2=19: LET N2=FN M(YR+1): LET K2=30: LET D2=FN M(C2+(N2*19))
680 IF N2>11 AND D2<27 THEN LET D2=D2-1: GOTO 700
690 IF N2<=11 AND D2=29 THEN LET D2=28
700 LET D2=D2+21
710 LET D2=D2+1: LET K2=7: IF INT (FN M(D2+DE)+0.1)<>1 THEN GOTO 710
720 IF D2<32 THEN  LET ME=3
730 IF D2>=32 THEN  LET D2=D2-31: LET ME=4
740 LET DE=INT (D2+0.1): LET MO=MS: LET DA=DS
750 RETURN
760 GOSUB 2510: GOSUB 2480
770 CLS
780 LET MK=5
790 CLS
800 PRINT AT 17,0;"<BREAK> retorna ao menu"
810 PRINT "Teclas z,x alteram Mes"
820 PRINT  INK 7;T$(1); INK 2;" F  "; INK 7;T$(2); INK 4;" A"
830 PRINT  INK 7;T$(3); INK 1;" C  "; INK 7;T$(4); INK 3;" F"
840 PRINT AT 0,0;
850 GOSUB 2570: IF MK<5 THEN PRINT T$(MK)
860 PRINT #P: LET KB=1: GOSUB 1920
870 IF P=3 THEN  PRINT #P
880 PRINT #P: LET T2=MK: LET S2=1: GOSUB 2020
890 LET P=2
900 LET K$="zxfacf ": GOSUB 1480: LET A=KB
910 IF A=1 THEN  LET MO=MO-1
920 IF A=2 THEN  LET MO=MO+1
930 IF MO=13 THEN  LET MO=1: LET YR=YR+1: GOSUB 640
940 IF MO=0 THEN  LET MO=12: LET YR=YR-1: GOSUB 640
950 IF A>2 AND A<7 THEN  LET MK=A-2
960 IF A<3 THEN  LET MK=5
970 IF A<>7 THEN  GOTO 790
980 RETURN
990 CLS : PRINT  PAPER 5; INK 1;AT 0,5;" CALENDARIO & DIARIO "; PAPER 6; INK 0;AT 2,7;" MENU PRINCIPAL "
1000 FOR Z=1 TO 19: PRINT  PAPER 1;"                                ": NEXT Z: PRINT AT 3,0
1010 PAPER 1: INK 7
1020 PRINT AT 4,1;"1- Consultar calendario mensal"
1030 PRINT AT 6,1;"2- Consultar calendario anual"
1040 PRINT AT 8,1;"3- Consultar diario"
```



```
1050 PRINT AT 10,1,"4- Rever/Ed
itar Financas"
1060 PRINT AT 12,1;"5- Rever/Ed
itar Apontamentos"
1070 PRINT AT 14,1;"6- Rever/Ed
itar Celebracoes"
1080 PRINT AT 16,1;"7- Rever/Ed
itar Feriados"
1090 PRINT AT 18,1;"8- Gravar a
s listas"
1100 PRINT AT 20,1;"9- Sair do
programa"
1110 PRINT TAB 9;"Faca a opcao"
```






```
10 CLS
20 CLEAR 5000
30 DIM LI$(3,150),TY$(3),CO(4)
40 DM$="31283130313031313031303
1"
50 MN$="JANEIRO  FEVEREIROMARCO
   ABRIL    MAIO      JUNHO
JULHO    AGOSTO   SETEMBRO OUTU
BRO  NOVEMBRO DEZEMBRO "
60 DN$="DOMSEGTERQUAQUISEXSAB"
70 PA$="MENSTRIMANUAUNIC"
80 TY$(0)="FINANCAS":TY$(1)="EN
CONTROS":TY$(2)="CELEBRACOES":T
Y$(3)="FERIADOS"
90 DEF FNM(A)=INT((A/K2-INT(A/K
2))*K2+0.5)*SGN(A/K2)
100 SV=0:P=0:MO=0:DA=0
110 PRINT @256,"HA ALGUMA LISTA
 DE DAOS GRAVADA? (S/N)"
120 REM
130 CLS:GOSUB 1030:CLS:P=0
140 IF C=1 GOSUB 770
150 IF C=2 GOSUB 2010
160 IF C=3 GOSUB 2460
170 IF C>3 AND C<8 THEN KB=C-4:
GOSUB 1180:SV=1
180 IF C=8 GOSUB 1730:SV=0
190 IF C=9 AND SV=1 THEN PRINT:
PRINT"VOCE NAO GRAVOU AS ALTERA
COES":PRINT"CONFIRMA A SAIDA?":
KB$="SN":GOSUB 1590:IF KB=2 THE
N C=0
200 IF C<>9 THEN 120
210 CLS:PRINT" ADEUS !"
220 END
230 'COMPRIMENTO DO MES
240 MX=0:A2=0
250 K2=4:IF FNM(YR)=0 THEN A2=1
260 K2=100:IF FNM(YR)=0 THEN A2
=0
270 K2=400:IF FNM(YR)=0 THEN A2
=1
280 IF KB=2 THEN MX=A2+28
290 IF KB<>2 THEN MX=VAL(MID$(D
M$,KB*2-1,2))
300 KB=MX:RETURN
310 'CARACTER DECISORIO
320 A3$="":N3=0:F3=0:M3=0
330 IF P=2 THEN RS=32:GOTO 460
340 IF KB=5 AND MO=ME AND DA=DE
 THEN RS=191:GOTO 460
350 IF KB=5 THEN RS=143:GOTO 46
0
360 M3=VAL(LI$(KB,0))
370 REM
380 N3=N3+1
390 A3$=LI$(KB,N3)
400 IF MID$(A3$,2,1)="" THEN D=
0 ELSE D=ASC(MID$(A3$,2,1))
410 IF D<>DA THEN 430
420 KB$=A3$:GOSUB 470:F3=K2
430 IF NOT(F3=1 OR N3>M3)THEN 3
70
440 IF F3=0 THEN RS=32:GOTO 460
450 N3=159+16*KB:RS=N3
460 KB=RS:RETURN
470 'CONFERIR ITEM NO MES
480 T4=0:Y4=0:F4=0
490 T4=ASC(MID$(KB$,1,1))
500 Y4=ASC(MID$(KB$,3,1))
510 M4=(Y4 AND 15)
520 Y4=(FIX(Y4/16)+17)*100+ASC(
MID$(KB$,4,1))
530 IF Y4>YR THEN K2=0:RETURN
540 K2=3:IF(T4=1 AND MO>=M4)OR(
T4=2 AND FNM(M4-MO)=0)OR(T4=3 A
ND M4=MO)OR(T4=4 AND M4=MO AND
Y4=YR) THEN F4=1
550 K2=F4:RETURN
560 'NUMERO DO DIA
570 YX=0:D2=0:M2=0
580 Y2=YR-1
590 D2=Y2*365+FIX(Y2/4)-FIX(Y2/
100)+FIX(Y2/400)
600 IF MO=1 THEN 640
610 FOR M2=1 TO MO-1
620 KB=M2:GOSUB 230:D2=D2+KB
630 NEXT
640 KB=D2+DA:RETURN
650 REM
660 N2=0:C2=0:D2=0
670 MS=MO:DS=DA
680 DA=1:MO=3:GOSUB 560:K2=7:DE
=FNM(KB)
690 N2=FIX(YR/100)-16:C2=3+N2-F
IX((N2+1)/3):FIX(N2/4)
700 K2=19:N2=FNM(YR+1):K2=30:D2
=FNM(C2+(N2*19))
710 IF N2>11 AND D2<27 THEN D2=
D2-1 ELSE IF N2<=11 AND D2=29 T
HEN D2=28
720 D2=D2+21
730 D2=D2+1:K2=7:IF FNM(D2+DE)<
>1 THEN 730
740 IF D2<32 THEN ME=3 ELSE D2=
D2-31:ME=4
750 DE=D2:MO=MS:DA=DS
760 RETURN
770 'VER CAL.MES
780 REM
790 GOSUB 2750:GOSUB 2720
800 CLS
810 PRINT"SETAS UP/DOWN ALTERAM
 MES"
820 PRINT"USE clear PARA VOLTAR
 AO MENU"
830 PRINT:PRINT CHR$(159);TY$(0
);"($)",CHR$(175);TY$(1)
840 PRINT:PRINT CHR$(191);TY$(2
),CHR$(207);TY$(3)
850 PRINT:PRINT"QUALQUER TECLA
PARA CONTINUAR"
855 IF INKEY$="" THEN 855
```

```
860 MK=5
870 REM
880 CLS
890 GOSUB 2820:IF MK<4 THEN PRI
NT @19,TY$(MK)
900 PRINT #-P:KB=1:GOSUB 2150
910 IF P=2 THEN PRINT #-P
920 PRINT#-P:T2=MK:S2=1:GOSUB 2
240
930 P=0
940 KB$="^"+CHR$(10)+"$ECF"+CHR
$(12):GOSUB 1590:A=KB
950 IF A=1 THEN MO=MO-1
960 IF A=2 THEN MO=MO+1
970 IF MO=13 THEN MO=1:YR=YR+1:
GOSUB 650
980 IF MO=0 THEN MO=12:YR=YR-1:
GOSUB 650
990 IF A>2 AND A<7 THEN MK=A-3
1000 IF A<3 THEN MK=5
1010 IF A<>7 THEN 870
1020 RETURN
1030 'RETORNO AO MENU
1040 PRINT " PROGRAMA  CALENDAR
IO & DIARIO  ";STRING$(32,131)
1050 PRINT TAB(13);"menu"
1070 PRINT"1- CONSULTAR CALENDA
RIO MENSAL"
1080 PRINT"2- CONSULTAR CALENDA
RIO ANUAL"
1090 PRINT"3- CONSULTAR DIARIO"
1100 PRINT"4- REVER/EDITAR FINA
NCAS"
1110 PRINT"5- REVER/EDITAR ENCO
NTROS"
1120 PRINT"6- REVER/EDITAR CELE
BRACOES"
1130 PRINT"7- REVER/EDITAR FERI
ADOS"
1140 PRINT"8- GRAVAR AS LISTAS"
1150 PRINT"9- SAIR DO PROGRAMA"
1160 PRINT:PRINT TAB(9);"FACA A
 OPCAO"
1170 KB$="123456789":GOSUB 1590
:C=KB:RETURN
```

```
10 CLS:COLOR 15,4,4:KEYOFF
12 CLEAR 7000:MAXFILES=3
15 OPEN "CRT:" FOR OUTPUT AS #3
20 OPEN "LPT:" FOR OUTPUT AS #2
30 DIM LI$(3,150),TY$(3),CO(4)
40 DM$="31283130313031313031303
1"
50 MN$="JANEIRO  FEVEREIROMARÇO
   ABRIL    MAIO     JUNHO
JULHO    AGOSTO   SETEMBRO OUTU
BRO  NOVEMBRO DEZEMBRO ":' O nú
mero de espaços deve completar
9 caracteres para cada mês
60 DN$="DOMSEGTERQUAQUISEXSAB"
70 PA$="menstrimanuaunic"
80 TY$(0)="Finanças":TY$(1)="En
contros":TY$(2)="Celebrações":T
Y$(3)="Feriados"
90 DEF FNM(A)=INT((A/K2-INT(A/K
2))*K2+.5)*SGN(A/K2)
100 SV=0:P=0:MO=0:DA=0
110 LOCATE 1,5:PRINT"Você tem l
istas de dados? (S/N)";
120 REM
130 CLS:GOSUB 1030:CLS:P=3
140 IF C=1 THEN GOSUB 770
150 IF C=2 THEN GOSUB 2010
160 IF C=3 THEN GOSUB 2460
170 IF C>3 AND C<8 THEN KB=C-4:
GOSUB 1180:SV=1
180 IF C=8 THEN GOSUB 1730:SV=0
190 IF C=9 AND SV=1 THEN PRINT:
PRINT"Você não gravou as ultima
s alterações!":PRINT"Confirma a
 saída? (S/N)";:KB$="SN":GOSUB
1590:IF KB=2 THEN C=0
200 IF C<>9 THEN 120
210 CLS:CLOSE:PRINT"Até logo...
"
220 END
230 ' Tamanho do mês
240 MX=0:A2=0
250 K2=4:IF FNM(YR)=0 THEN A2=1
260 K2=100:IF FNM(YR)=0 THEN A2
=0
270 K2=400:IF FNM(YR)=0 THEN A2
=1
280 IF KB=2 THEN MX=A2+28
290 IF KB<>2 THEN MX=VAL(MID$(D
M$,KB*2-1,2))
300 KB=MX:RETURN
310 ' Marcar dia de compromisso
320 A3$="":N3=0:F3=0:M3=0
330 IF P=2 THEN RS=32:GOTO 460
340 IF KB=5 AND MO=ME AND DA=DE
 THEN RS=42:GOTO 460
350 IF KB=5 THEN RS=32:GOTO 460
360 M3=VAL(LI$(KB,0))
370 REM
380 N3=N3+1
390 A3$=LI$(KB,N3)
400 IF MID$(A3$,2,1)="" THEN D=
0 ELSE D=ASC(MID$(A3$,2,1))
410 IF D<>DA THEN 430
420 KB$=A3$:GOSUB 470:F3=K2
430 IF NOT (F3=1 OR N3>M3) THEN
 370
440 IF F3=0 THEN RS=32:GOTO 460
450 RS=62
460 KB=RS:RETURN
470 ' Procura por ítem em mês
480 T4=0:Y4=0:F4=0
490 T4=ASC(MID$(KB$,1,1))
500 Y4=ASC(MID$(KB$,3,1))
510 M4=(Y4 AND 15)
520 Y4=(FIX(Y4/16)+17)*100+ASC(
MID$(KB$,4,1))
530 IF Y4>YR THEN K2=0:RETURN
540 K2=3:IF (T4=1 AND MO>=M4) O
R (T4=2 AND FNM(M4-MO)=0) OR (T
4=3 AND M4=MO) OR (T4=4 AND M4=
MO AND Y4=YR) THEN F4=1
550 K2=F4:RETURN
560 ' Det numero do dia
570 YX=0:D2=0:M2=0
580 Y2=YR-1
590 D2=Y2*365+FIX(Y2/4)-FIX(Y2/
100)+FIX(Y2/400)
600 IF MO=1 THEN 640
```



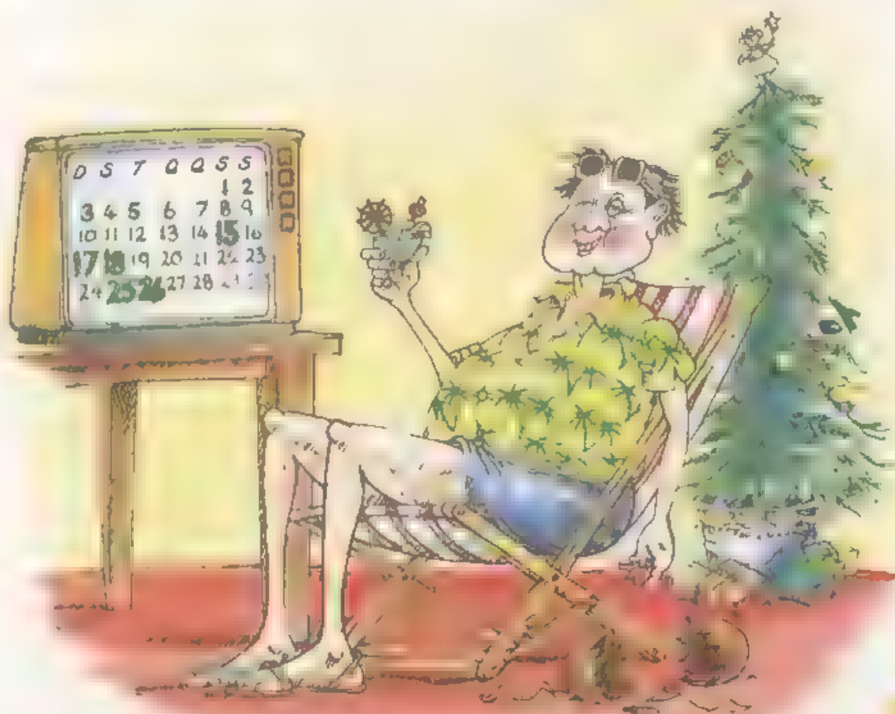
```
610 FOR M2=1 TO MO-1
620 KB=M2:GOSUB 230:D2=D2+KB
630 NEXT
640 KB=D2+DA:RETURN
650 ' Encontrar o dia de Páscoa
660 N2=0:C2=0:D2=0
670 MS=MO:DS=DA
680 DA=1:MO=3·GOSUB 560:K2=7:DE
=FNM(KB)
690 N2=FIX(YR/100)-16:C2=3+N2-F
IX((N2+1)/3)-FIX(N2/4)
700 K2=19:N2=FNM(YR+1):K2=30:D2
=FNM(C2+(N2*19))
710 IF N2>11 AND D2<27 THEN D2=
D2-1 ELSE IF N2<=11 AND D2=29 T
HEN D2=28
720 D2=D2+21
730 D2=D2+1:K2=7:IF FNM(D2+DE)<
>1 THEN 730
740 IF D2<32 THEN ME=3 ELSE D2=
D2-31·ME=4
750 DE=D2:MO=MS:DA=DS
760 RETURN
770 ' Consulta calend mensal
780 REM
790 GOSUB 2750:GOSUB 2720:MK=5
800 CLS:LOCATE 0,20
810 PRINT"<- e -> mudam o mês "
;
820 PRINT"<esc> volta ao menu";
830 PRINT"S:";TY$(0),"E:";TY$(1
)
840 PRINT"C:";TY$(2),"F:";TY$(3
)
850 LOCATE 0,0
860 REM
870 REM
880 REM
890 GOSUB 2820:IF MK<4 THEN LOC
ATE 3,2:PRINTTY$(MK)
895 REMIF P=2 THEN LPRINT
900 PRINT#P,:KB=1:GOSUB 2150
910 IF P=2 THEN LPRINT
920 PRINT#P,:T2=MK:S2=1:GOSUB 2
240
930 P=3
940 KB$=CHR$(29)+CHR$(28)+"$ECF
"+CHR$(27)·GOSUB 1590:A=KB
950 IF A=1 THEN MO=MO-1
960 IF A=2 THEN MO=MO+1
970 IF MO=13 THEN MO=1:YR=YR+1:
GOSUB 650
980 IF MO=0 THEN MO=12·YR=YR-1:
GOSUB 650
990 IF A>2 AND A<7 THEN MK=A-3
1000 IF A<3 THEN MK=5
1010 IF A<>7 THEN 800
1020 RETURN
1030 ' Menu devolve escolha em
KB
1040 LOCATE10,0:PRINT"CALENDARI
O E DIARIO"
1050 PRINT:PRINTTAB(16);"Menu
1060 PRINT
1070 PRINT"1- Consultar calendá
rio mensal"
1080 PRINT·PRINT"2- Consultar c
alendário anual"
1090 PRINT:PRINT"3- Consultar d
iário"
1100 PRINT:PRINT"4- Rever/edita
r finanças"
1110 PRINT:PRINT"5- Rever/edita
r encontros"
1120 PRINT:PRINT"6- Rever/edita
r celebrações"
1130 PRINT:PRINT"7- Rever/edita
r feriados"
1140 PRINT:PRINT"8- Gravar as l
istas"
1150 PRINT:PRINT"9- Fim de prog
rama"
1160 PRINT:PRINTTAB(9);"Escolha
:";
1170 KB$="123456789":GOSUB 1590
:C=KB:RETURN
```

```
10  HOME
30  DIM LI$(3,150),TY$(3),CO(4)

40 DM$ = "312831303130313130313
031"
50 MN$ = "JANEIRO  FEVEREIROMAR
CO    ABRIL    MAIO     JUNHO
  JULHO    AGOSTO   SETEMBRO OU
TUBRO  NOVEMBRO DEZEMBRO "
60 DN$ = "DOMSEGTERQUAQUISEXSAB
"
70 PA$ = "MENSALTRIMESANUAL UNI
CO ":D$ =  CHR$ (13) +  CHR$ (4
)
80 TY$(0) = "FINANCAS":TY$(1) =
 "ENCONTROS":TY$(2) = "CELEBRAC
OES":TY$(3) = "FERIADOS"
90  DEF  FN M(A) =  INT ((A / K
2 -  INT (A / K2)) * K2 + .5) *
 SGN (A / K2)
100 SV = 0:P = 0:MO = 0:DA = 0
110  VTAB 8: PRINT "VOCE TEM LI
STAS DE DADOS? (S/N) ";
120  REM
130  HOME : GOSUB 1030: HOME :P
 = 0
140  IF C = 1 THEN  GOSUB 770
150  IF C = 2 THEN  GOSUB 2010
160  IF C = 3 THEN  GOSUB 2460
170  IF C > 3 AND C < 8 THEN KB
 = C - 4: GOSUB 1180:SV = 1
180  IF C = 8 THEN  GOSUB 1730:
SV = 0
190  IF C = 9 AND SV = 1 THEN
PRINT : PRINT "VOCE NAO GRAVOU
AS ALTERACOES!": PRINT "CONFIRM
A A SAIDA? (S/N)":KB$ = "SN": G
OSUB 1590: IF KB = 2 THEN C = 0

200  IF C <  > 9 THEN 120
210  HOME : PRINT "ATE LOGO..."

220  END
230  REM  DURACAO DO MES
240 MX = 0:A2 = 0
250 K2 = 4: IF  FN M(YR) = 0 TH
EN A2 = 1
260 K2 = 100: IF  FN M(YR) = 0
THEN A2 = 0
270 K2 = 400: IF  FN M(YR) = 0
THEN A2 = 1
280  IF KB = 2 THEN MX = A2 + 2
8
290  IF KB <  > 2 THEN MX =  VA
L ( MID$ (DM$,KB * 2 - 1,2))
300 KB = MX: RETURN
310  REM  CARACTER MARCADOR
320 A3$ = "":N3 = 0:F3 = 0:M3 =
 0
330  IF P = 2 THEN RS = 32: GOT
O 460
340  IF KB = 5 AND MO = ME AND
DA = DE THEN RS = 42: GOTO 460
350  IF KB = 5 THEN RS = 32: GO
TO 460
```



```
360 M3 =  VAL (LI$(KB,0))
370  REM
380 N3 = N3 + 1
390 A3$ = LI$(KB,N3)
400  IF  MID$ (A3$,2,1) = "" TH
EN D = 0
405  IF  MID$ (A3$,2,1) <  > ""
 THEN D =  ASC ( MID$ (A3$,2,1)
)
410  IF (D <  > DA) THEN 430
420 KB$ = A3$: GOSUB 470:F3 = K
2
430  IF  NOT (F3 = 1 OR N3 > M3
) THEN 370
440  IF F3 = 0 THEN RS = 32: GO
TO 460
450 RS = 62
460 KB = RS: RETURN
470  REM  VERIFICA ITEM EM MES
480 T4 = 0:Y4 = 0:F4 = 0
490 T4 =  ASC ( MID$ (KB$,1,1))

500 Y4 =  ASC ( MID$ (KB$,3,1))
:T = Y4
510 Y4 = ( INT ( ABS (Y4 / 16))
 + 17) * 100 +  ASC ( MID$ (KB$
,4,1))
520 M4 = T - (( INT (Y4 / 100)
  17) * 16)
530  IF Y4 > YR THEN K2 = 0: RE
TURN
540 K2 = 3: IF (T4 = 1 AND ((Y4
 < YR) OR (MO >  = M4 AND Y4 =
YR))) OR (T4 = 2 AND  FN M(M4 -
 MO) = 0) OR (T4 = 3 AND M4 = M
O) OR (T4 = 4 AND M4 = MO AND Y
4 = YR) THEN F4 = 1
550 K2 = F4: RETURN
560  REM  NUMERO DO DIA
570 YX = 0:D2 = 0:M2 = 0
580 Y2 = YR - 1
590 D2 = Y2 * 365 +  INT ( ABS
(Y2 / 4)) -  INT ( ABS (Y2 / 10
0)) +  INT ( ABS (Y2 / 400))
600  IF MO = 1 THEN 640
610  FOR M2 = 1 TO MO - 1
620 KB = M2: GOSUB 230:D2 = D2
+ KB
630  NEXT
640 KB = D2 + DA: RETURN
650  REM  DATA DA PASCOA
660 N2 = 0:C2 = 0:D2 = 0
670 MS = MO:DS = DA
680 DA = 1:MO = 3: GOSUB 560:K2
 = 7:DE =  FN M(KB)
690 N2 =  INT ( ABS (YR / 100))
 - 16:C2 = 3 + N2 -  INT ( ABS
((N2 + 1) / 3)) -  INT ( ABS (N
2 / 4))
700 K2 = 19:N2 =  FN M(YR + 1):
K2 = 30:D2 =  FN M(C2 + (N2 * 1
9))
710  IF N2 > 11 AND D2 < 27 THE
N D2 = D2 - 1: GOTO 720
715  IF N2 <  = 11 AND D2 = 29
THEN D2 = 28
720 D2 = D2 + 21
730 D2 = D2 + 1:K2 = 7: IF  FN
M(D2 + DE) <  > 1 THEN 730
740  IF D2 < 32 THEN ME = 3: GO
TO 750
745 D2 = D2 - 31:ME = 4
750 DE = D2:MO = MS:DA = DS
760  RETURN
770  REM  CALENDARIO MENSAL
780  REM
790  GOSUB 2750: GOSUB 2720:MK
= 5
800  HOME
810  VTAB 21: PRINT "<- ->: MUD
A O MES     ";
820  PRINT "<ESC> = MENU"
830  PRINT "S:"TY$(0),"E:"TY$(1
)
840  PRINT "C:"TY$(2),"F:"TY$(3
);
850  REM
855  REM
870  REM
880  PRINT D$;"PR#";P: VTAB 1:
HTAB 1
890  GOSUB 2820: IF MK < 4 THEN
  HTAB 20: VTAB 1: PRINT TY$(MK
)
900  PRINT : PRINT :KB = 1: GOS
UB 2150
910  IF P = 1 THEN  PRINT
920  PRINT :T2 = MK:S2 = 1: GOS
UB 2240
930 P = 0: PRINT D$;"PR#";P
940 KB$ =  CHR$ (8) +  CHR$ (21
) + "SECF" +  CHR$ (27): GOSUB
1590:A = KB
950  IF A = 1 THEN MO = MO - 1
960  IF A = 2 THEN MO = MO + 1
970  IF MO = 13 THEN MO = 1:YR
= YR + 1: GOSUB 650
980  IF MO = 0 THEN MO = 12:YR
= YR - 1: GOSUB 650
990  IF A > 2 AND A < 7 THEN MK
 = A - 3
1000  IF A < 3 THEN MK = 5
1010  IF A <  > 7 THEN 800
1020  RETURN
1030  REM  MENU DEVOLVE ESCOLHA
 EM KB
1040  INVERSE : PRINT  TAB( 4)"
PROGRAMA DE CALENDARIO E DIARIO
    ": NORMAL
1050  PRINT : PRINT  TAB( 18)"M
ENU"
1060  PRINT
1070  PRINT  TAB( 8);"1:-VER CA
LENDARIO MENSAL"
1080  PRINT : PRINT  TAB( 8);"2
:-VER CALENDARIO ANUAL"
1090  PRINT : PRINT  TAB( 8);"3
:-VER DIARIO"
1100  PRINT : PRINT  TAB( 8);"4
:-REVER/EDITAR FINANCAS"
1110  PRINT : PRINT  TAB( 8);"5
: REVER/EDITAR ENCONTROS"
1120  PRINT : PRINT  TAB( 8);"6
:-REVER/EDITAR CELEBRACOES"
1130  PRINT : PRINT  TAB( 8);"7
:-REVER/EDITAR FERIADOS"
1140  PRINT : PRINT  TAB( 8);"8
:-GRAVAR AS LISTAS"
1150  PRINT : PRINT  TAB( 8);"9
:-SAIR DO PROGRAMA"
1160  PRINT : PRINT : PRINT  TA
B( 15)"ESCOLHA ";
1170 KB$ = "123456789": GOSUB 1
590:C = KB: RETURN
```

MICRO DICAS

COMO CALCULAR DATAS EM UM PROGRAMA DE CALENDÁRIO

Um problema sério de programação, para quem deseja desenvolver aplicações que envolvem cálculos de datas ou a determinação das funções de um calendário universal, é a realização desses cálculos em BASIC ou outra linguagem de alto nível.

Há várias soluções para o problema — umas mais fáceis e intuitivas e outras mais difíceis, mas que utilizam algoritmos ("truques" matemáticos de cálculo) bem compactos. Desenvolvido o procedimento para todos os tipos de cálculo, organize-os na forma de sub-rotinas, que poderão ser usadas em diversos programas.

Esse assunto será tratado em um artigo especial, mas adiantamos aqui algumas considerações importantes.

• Teste de datas

Tenha sempre o cuidado de testar a data fornecida pelo usuário, que deverá ser entrada no formato numérico (no Brasil, DD/MM/AAAA).

Verifique se o número do mês cai entre 01 e 12 e se o dia do mês cai entre 1 e o número de dias no mês (28, 29, 30 ou 31). Pode-se armazenar o número de dias para cada mês em um conjunto de doze elementos.

Teste, também, se o ano é bissexto. Para isso, basta verificar se ele é divisível exatamente por 4.

A maneira de testar o ano varia com a aplicação. Se se pede ao usuário a data atual, por exemplo, pode-se incluir na rotina um teste que indique se ela é ou não anterior à data de criação do programa!

• Tipos de data

O tipo de data que utilizamos (DD/MM/AA), chamado de data gregoriana, é muito inconveniente para entrada em computador e para cálculos de diferenças de datas. Para colocá-las em ordem crescente é preciso alterar a ordem dos seus três elementos: o algoritmo de ordenação deve testá-las como AAMMDD.

Existem outros tipos de data, mais racionais. A data ordinal, por exemplo, torna bem mais fácil calcular diferenças entre datas. É expressa numa forma como esta: 123/1986 — ou seja, dia n.º 123 (desde 1.º de janeiro) do ano de 1986. A data fiscal, por sua vez, leva em consideração o número do dia da semana e o número da semana (3/25/1986 é a terça-feira da 25.ª semana de 1986).

UMA AGENDA ELETRÔNICA (2)

- COMO USAR O PROGRAMA
- CONSULTE O CALENDÁRIO
- CONSULTE A AGENDA
- DIGITAÇÃO DOS PRIMEIROS DADOS
- VERIFIQUE AS LISTAS

Festividades, aniversários, encontros, reuniões... Você costuma esquecer-se dos compromissos? Anime-se: com a agenda e o calendário eletrônicos você nunca mais passará vergonha.

Carregado na memória o programa da lição anterior, digite as linhas aqui apresentadas. O computador perguntará se alguma lista de dados já foi gravada. Como isso ainda não aconteceu, responda N. O programa mostrará então o menu principal com nove opções:

1. Consultar o calendário mensal
2. Consultar o calendário anual
3. Consultar a agenda
4. Rever/editar finanças
5. Rever/editar encontros
6. Rever/editar celebrações
7. Rever/editar feriados
8. Gravar as listas
9. Sair do programa

Pode-se consultar o calendário sem digitar nenhum dado previamente. Vamos começar por ele.

CONSULTE O CALENDÁRIO

Escolha a opção 1 e será possível ver o calendário de qualquer mês dentro dos limites do programa. A lista de meses é, na realidade, bastante grande. Ela vai de 1753, quando o calendário gregoriano foi adotado na Grã-Bretanha, até 29999. O mês deve ser sempre escolhido pelo número (de 1 a 12) e não pelo nome.

Se você dispuser de uma impressora, pode optar por trabalhar com uma cópia impressa do calendário. Caso contrário, ele será apresentado apenas na tela. O nome do mês, bem como o ano, é mostrado no topo da tela, com os dias logo abaixo. A semana começa com o domingo, que é colocado na cor vermelha em alguns computadores. A data da Páscoa será calculada automaticamente e mostrada no rodapé, se cair no mês escolhido (março ou abril).

Muitas coisas podem ser feitas durante a consulta ao calendário mensal. A lista de opções aparece no rodapé da tela ou na página anterior, conforme o micro. Teclar <BREAK>, <ESCAPE> ou <CLEAR> levará você de volta ao menu principal. Outras teclas permitem-lhe avançar ou retroceder de mês em mês. O Spectrum usa as teclas Z e X; o TRS-Color, as setas para cima e para baixo; e o Apple e o MSX, as setas para a esquerda e para a direita.

As outras teclas com funções no programa são as seguintes: **$**, que serve para destacar dados referentes a finanças; **E**, utilizada para encontros; **C**, para celebrações, e **F**, para feriados. Mas nada acontecerá até que alguns dados sejam introduzidos no computador.

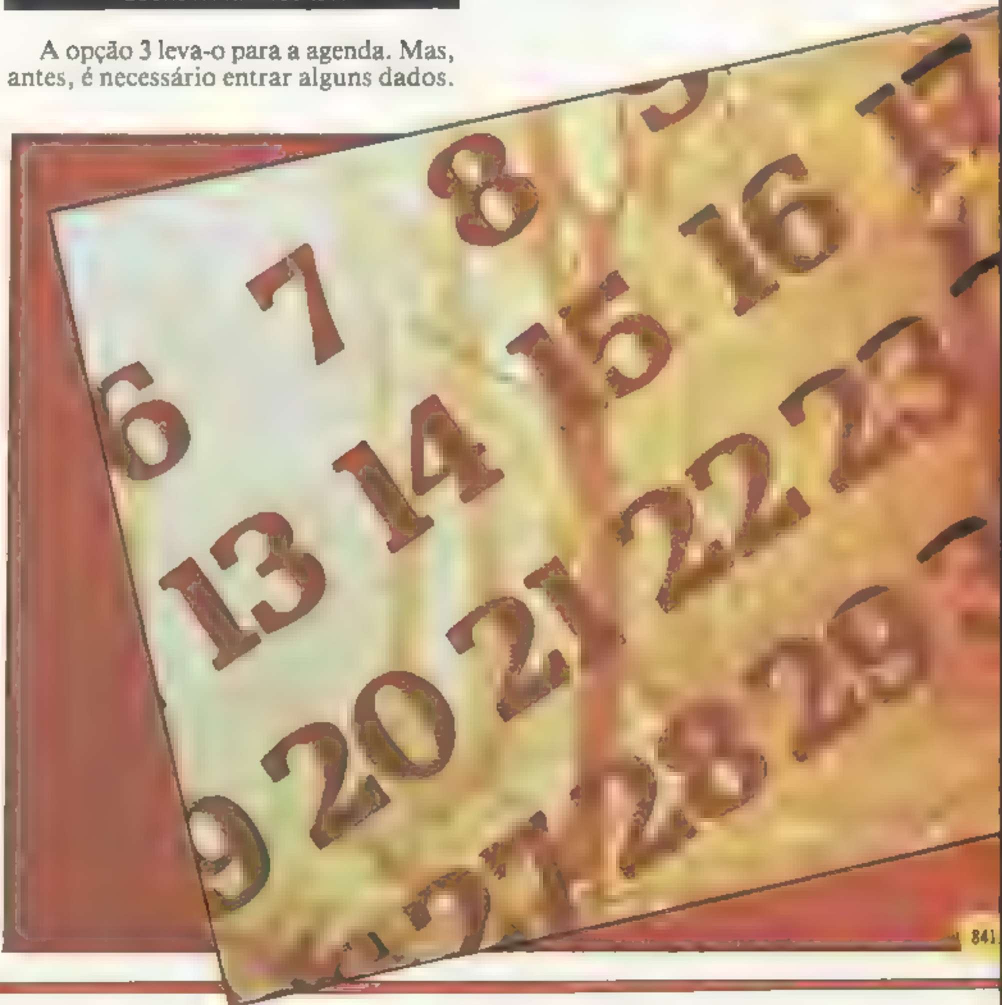
ESCREVA NA AGENDA

A opção 3 leva-o para a agenda. Mas, antes, é necessário entrar alguns dados. Assim, escolha a opção adequada a partir do menu principal.

Qualquer que seja a opção, tecle A para adicionar um dado, D para apagar e M para retornar ao menu.

Como já foi descrito, a seção de finanças oferece as opções Mensal, Trimestral, Anual ou entrada única. Faça a opção teclando a letra inicial correspondente (M, T, A ou U). Em seguida, o programa pede que você entre um nome ou frase com um máximo de vinte letras, e que descreva o evento com sua respectiva data.

Para um dado recorrente, esta seria a primeira vez que ele acontece (o primeiro aluguel pago, por exemplo). O programa completa automaticamente os meses seguintes com os dados adequados. Apontamentos e feriados são tra-



tados como eventos únicos, e celebrações, como eventos anuais. Use esta opção para comemorações como aniversários de nascimento e de casamento etc.

Pode-se executar até 150 entradas em cada categoria.

Mas, para fazer tudo isso, será preciso esperar as rotinas do proximo artigo Esse artigo apresentara tambem a parte final do programa, que permitirá gravar, carregar e imprimir dados, concluindo o processamento.

```
165 GOSUB 1480· IF KB=1 THEN
 GOSUB 1690
1120 PAPER 0· INK 7
1130 LET K$="123456789"· GOSUB
1480  LET C=KB: RETURN
1140 LET KK=KB
1150 LET KB=KK· GOSUB 1330
1160 LET B=Q(KK): FOR y=4 TO 20
 PRINT AT y,0;Z$· NEXT y· PRIN
T AT 4,0
1170 IF B=0 THEN  GOTO 1210
1180 FOR N=1 TO B
1190 LET K$=L$(KK,N)· LET K2=N·
 GOSUB 1370
1200 NEXT N
1210 LET K$="adm"  GOSUB 1480·
LET A=KB
1220 FOR I=1 TO 100· NEXT I
1230 IF A<>1 THEN  GOTO 1280
1240 LET K2=B: LET T7=KK: GOSUB
 1550· LET V$=STR$ T7· GOSUB 14
00
1250 LET W$=STR$ DA  IF LEN W$=
1 THEN  LET W$="0"+W$
1260 LET V$ V$+W$· LET W$=STR$
MO  IF LEN W$=1 THEN  LET W$=
"0 "+W$
1270 LET V$=V$+W$+STR$ YR·
 LET  L$(KK,B+1)=V$+B$  LET Q
(KK)=Q(K K +1
1280 IF A<>2 THEN  GOTO 1310
1290 INPUT "APAGAR QUAL
NUMERO   (MIN 1)";NN  IF NN<1
OR NN>B THEN  GOTO 1290
1300 FOR Z=NN+1 TO B  LET
L$(KK ,Z-1)=L$(KK,Z)· NEXT
Z· LET Q(K K) =Q(KK)-1
1310 IF A<>3 THEN  GOTO 1160
1320 RETURN
1330 PRINT "Lista de Dados"
'T$( KB)
1340 PRINT AT 0,17; INVER
SE 1," S"; INVERSE 0;"OMAR
"; INVERSE  1,"A"; INVERSE
0;"PAGAR "; INVERSE 1,"M";
INVERSE 0;"ENU"
1350 PRINT AT 3,0;
1360 RETURN
1370 LET F$=L$(KK,K2):
LET E$=P $(VAL F$( TO 1))
+""+F$(2 TO 3)+ ":"+F$(4
TO 5)+" "+F$(6 TO 9)
1380 PRINT E$;" ";F$(10
TO 30)
1390 RETURN
1400 LET B$=""· LET VP=5
1410 INPUT "Qual o nome ? (Max
22 letras)"' LINE B$
1420 LET VP=VP-1
1430 PRINT AT VP,0;"";
1440 INPUT "DATA SIGNIFICATIVA
·",DA: IF DA<1 OR DA>31 THEN  G
OTO 1430
1450 GOSUB 2510
1460 LET KB=MO: GOSUB 270. IF D
A>KB THEN  GOTO 1430
1470 LET K$=B$: GOSUB 1520: LET
 Y$=K$· RETURN
1480 LET a$=INKEY$: IF a$="" TH
EN  GOTO 1480
1490 FOR n=1 TO LEN K$: IF a$<>
K$(n) THEN  NEXT n
1500 IF n>LEN K$ THEN  GOTO 148
0
1510 LET KB=n: RETURN
1520 LET PP=(INT (YR/100)-17)*1
6+MO
1530 LET K2=100: LET QQ=FN M(YR
)
1540 LET K$=CHR$ PP+CHR$ QQ+K$:
 RETURN
1550 IF T7<>1 THEN  GOTO 1580
1560 PRINT AT 20,0; INVERSE 1,"
M"; INVERSE 0;"ENSAL "; INVERSE
 1;"T"; INVERSE 0;"RIMESTRAL ";
 INVERSE 1;"A"; INVERSE 0;"NUAL
"; INVERSE 1;"U"; INVERSE 0,"N
ICO"
1570 LET K$="mtau": GOSUB 1480:
 PRINT AT 20,0;Z$: PRINT AT 3,0
;: LET T7=KB: GOTO 1600
1580 IF T7=3 THEN  RETURN
1590 LET T7=4
1600 RETURN
1610 CLS · PRINT  FLASH 1;AT 10
,5;"PREPARE O GRAVADOR"
1620 IF INKEY$<>"" THEN  GOTO 1
620
1630 SAVE "dados" DATA Q()
1640 FOR N=1 TO 4
1650 IF Q(N)=0 THEN  GOTO 1670
1660 FOR M=1 TO Q(N)· LET O$(1)
=L$(N,M). SAVE "dados" DATA O$(
): NEXT M
1670 NEXT N
1680 RETURN
1690 CLS · PRINT AT 10,5; FLASH
 1;"PREPARE O GRAVADOR"
1700 LOAD "dados" DATA Q()
1710 FOR N=1 TO 4
1720 IF Q(N)=0 THEN  GOTO 1740
1730 FOR M=1 TO Q(N)· LOAD "dad
os" DATA O$()· LET L$(N,M)=O$(1
```



```
)· NEXT M
1740 NEXT N
1750 RETURN
```

```
115 KB$="SN":GOSUB 1590:IF KB=1
 GOSUB 1870
1180 REM
1190 N=0:A=0:B=0:KK=KB
1200 KB=KK:GOSUB 1330
1210 REM
1220 FOR VU=1 TO 14:PRINT @VU*3
2:NEXT:PRINT @32,"";
1230 B=VAL(LI$(KK,0))
1240 IF B=0 THEN 1280
1250 FOR N=1 TO B
1260 KB$=LI$(KK,N):K2=N:GOSUB 1
410
1270 NEXT
1280 KB$="ADM"·GOSUB 1590·A=KB
1290 IF A=1 THEN T7=KK·GOSUB 16
80.GOSUB 1490·LI$(KK,B+1)=CHR$(
T7)+T8$:LI$(KK,0)=MID$(STR$(B+1
),2)
1300 IF A=2 THEN INPUT"QUE NUME
RO";NN·IF NN<1 OR NN>B THEN 130
0 ELSE FOR PP=NN+1 TO B:LI$(KK,
PP-1)=LI$(KK,PP)·NEXT:LI$(KK,0)
=STR$(B-1)
1310 IF A<>3 THEN 1210
1320 RETURN
1330 REM
1340 PRINT TY$(KB),"LISTA CORRE
NTE"
1350 FOR Y=2 TO 14
1360 PRINT @Y*32
1370 NEXT
1380 PRINT @480,"aDICIONAR dELE
TAR mENU";
1390 PRINT @32,"";
1400 RETURN
1410 'OP
1420 N2=0.BB$="":DD$=""·K3=K2
1430 FOR N2=1 TO 4:IF MID$(KB$,
N2,1)="" THEN CO(N2)=0 ELSE CO(
N2)=ASC(MID$(KB$,N2,1))
1440 NEXT
1450 BB$=MID$(PA$,CO(1)*4-3,4)
1460 K2=16:DD$=MID$(STR$(CO(2))
,2)+":"+MID$(STR$(FNM(CO(3))),2
)+":"+MID$(STR$((FIX(CO(3)/16)+
17)*100+CO(4)),2)
1470 PRINT MID$(STR$(K3),2);" "
;BB$;" ";DD$;" ";RIGHT$(KB$,LEN
(KB$)-4)
1480 RETURN
1490 'ADICIONAR UMA ENTRADA
1500 B3$="":VP=0
1510 PRINT"A SER CHAMADO? (MAX
22 LETRAS)":LINE INPUT B3$
1520 VP=INT((PEEK(136)*256+PEEK
(137)-1024)/32)
1530 PRINT @VP*32,"";
1540 PRINT"DATA SIGNIFICATIVA
? "
1550 INPUT"  DIA:";DA:IF DA<1
 OR DA>31 THEN 1530
1560 GOSUB 2750
1570 KB=MO:GOSUB 230:IF DA>KB
 THEN 1530
1580 KB$=B3$:GOSUB 1630:T8$
=KB$  :RETURN
1590 REM
1600 B$=INKEY$:IF B$=""
 THEN  1600
1610 KB=INSTR(1,KB$,B$)
 IF KB=0  THEN 1600
1620 RETURN
1630 REM
1640 PP=0·QQ=0
1650 PP=(FIX(YR/100)-17)
*16+MO
1660 K2=100.QQ=FNM(YR)
1670 KB$=CHR$(DA)+CHR$(
PP)+CHR$  (QQ)+LEFT$(KB$
,22):RETURN
1680 REM
1690 IF T7=0 THEN PRINT
"MENSAL, TRIMESTRAL,ANU
AL OU UNICO":KB$= "MTAU
":GOSUB 1590:T7=KB:GOTO
17  20
1700 IF T7=2 THEN T7=3:
GOTO 1720
1710 T7=4
1720 RETURN
1730 REM
1740 N=0:P=0
1750 OPEN"O",#-1,"DIARIO"
1760 FOR N=0 TO 3
1770 M=VAL(LI$(N,0))
1780 PRINT #-1,LI$(N,0)
1790 IF M=0 THEN 1840
1800 FOR P=1 TO M
1810 FOR J=1 TO 4:PRINT#-1,STR$
(ASC(MID$(LI$(N,P),J,1)))·NEXT J
1820 PRINT #-1,MID$(LI$(N,P),5)
1830 NEXT P
1840 NEXT N
1850 CLOSE #-1
1860 RETURN
1870 REM
1880 N=0:P=0:M=0
1890 OPEN"I",#-1,"DIARIO"
1900 FOR N=0 TO 3
1910 LINE INPUT #-1,LI$(N,0)
1920 M=VAL(LI$(N,0))
1930 IF M=0 THEN 1980
1940 FOR P=1 TO M
1950 FOR J=1 TO 4:INPUT#-1,NN$:
LI$(N,P)=LI$(N,P)+CHR$(VAL(NN$)
):NEXT J
1960 LINE INPUT #-1,NN$ LI$(N,P
)=LI$(N,P)+NN$
1970 NEXT
1980 NEXT
1990 CLOSE #-1
2000 RETURN
```

```
2010 'CAL.ANUAL
2020 M4=0:A4=0
2030 INPUT"ANO:";YR:IF YR<1753
OR YR>29999 THEN 2030 ELSE GOSU
B 650
2040 GOSUB 2720:CLS
2050 PRINT"ANO ";YR
2060 IF P=2 THEN PRINT #-2,"ANO
 ";YR
2070 PRINT #-P:KB=0:GOSUB 2150:
PRINT#-P
2080 GOSUB 2660
2090 FOR MO=1 TO 12
2100 PRINT #-P,MID$(MN$,MO*9-8,
9)
2110 T2=5:S2=0:GOSUB 2240
2120 IF P=0 AND INKEY$="" THEN
2120
2130 NEXT
2140 RETURN
2150 REM
2160 X2=0:C2=0:D2=0
2170 IF KB=0 THEN X2=7
2180 IF P=2 THEN KB=KB+1
2190 PRINT #-P,STRING$(X2,32);
2200 FOR D2=0 TO 6
2210 PRINT #-P,STRING$(KB," ")+
MID$(DN$,D2*3+1,3);
2220 NEXT
2230 RETURN
```



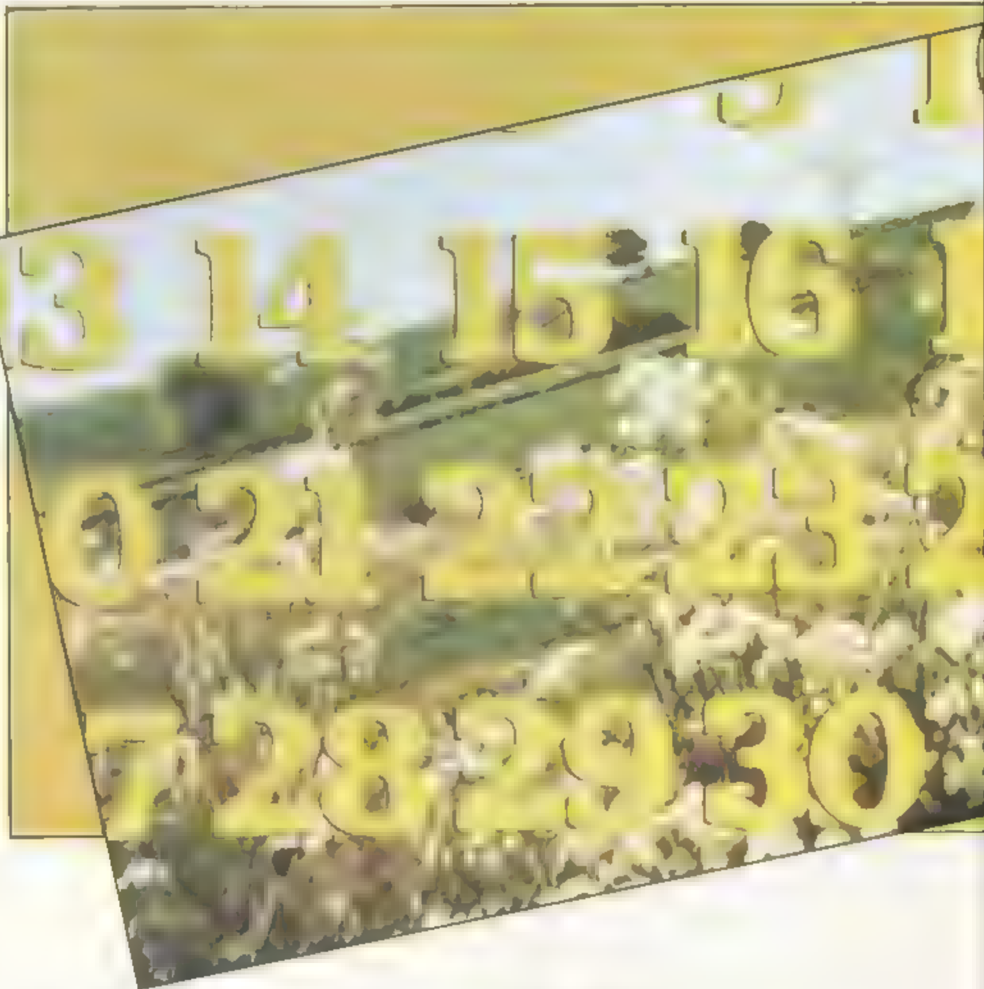
```
115 KB$="SN":GOSUB 1590:IF KB=1
 THEN GOSUB 1870
1180 ' LISTAR E ATUALIZAR
1190 N=0:A=0:B=0:KK=KB
1200 KB=KK:GOSUB 1330
1210 REM
1220 FOR VU=1 TO 21:LOCATE 0,VU
:PRINTSPACE$(38);:NEXT:LOCATE 0
,1
1230 B=VAL(LI$(KK,0))
1240 IF B=0 THEN 1280
1250 FOR N=1 TO B
1260 KB$=LI$(KK,N):K2=N:GOSUB 1
410
1270 NEXT
1280 KB$="ADM":GOSUB 1590:A=KB
1290 IF A=1 THEN T7=KK:GOSUB 16
80:GOSUB 1490:LI$(KK,B+1)=CHR$(
T7)+T8$:LI$(KK,0)=MID$(STR$(B+1
),2)
1300 IF A=2 THEN INPUT "QUAL NU
MERO",NN:IF NN <1 OR NN>B THEN
1300 ELSE FOR PP=NN+1 TO B:LI$(
KK,PP-1)=LI$(KK,PP):NEXT:LI$(KK
,0)=STR$(B-1)
1310 IF A<>3 THEN 1210
1320 RETURN
1330 ' Imprime cabeçalho
1340 PRINTTY$(KB),"Lista de dados"
1350 FOR Y=2 TO 21
1360 LOCATE 0,Y:PRINTSPACE$(38);
1370 NEXT
1380 LOCATE 5,23:PRINT"Adiciona
r Deletar Menu";
1390 LOCATE 0,1
1400 RETURN
1410 ' OP
1420 N2=0:BB$="":DD$="":K3=K2
1430 FOR N2=1 TO 4:IF MID$(KB$,
N2,1)="" THEN CO(N2)=0 ELSE CO(
N2)=ASC(MID$(KB$,N2,1))
1440 NEXT
1450 BB$=MID$(PA$,CO(1)*4-3,4)
1460 K2=16:DD$=MID$(STR$(CO(2))
,2)+":"+MID$(STR$(FNM(CO(3))),2
)+":"+MID$(STR$((FIX(CO(3)/16)+
17)*100+CO(4)),2)
1470 PRINTMID$(STR$(K3),2);" ";
BB$;" ";DD$;" ";RIGHT$(KB$,LEN(
KB$)-4)
1480 RETURN
1490 ' Adiciona um dado
1500 B3$="":VP=0
1510 PRINT"Rótulo? (max 20 letr
as)":LINEINPUT B3$
1520 PRINT
1530 REM
1540 PRINT"Data do compromisso:
"
1550 INPUT" Dia:";DA:IF DA<1 OR
 DA>31 THEN 1530
1560 GOSUB 2750
1570 KB=MO:GOSUB 230:IF DA>KB T
HEN 1530
1580 KB$=B3$:GOSUB 1630:T8$=KB$
:RETURN
1590 ' Verif teclagem de caract
er e compara com kb$
1600 B$=INKEY$:IF B$="" THEN 16
00
1610 KB=INSTR(KB$,B$):IF KB=0 T
HEN 1600
1620 RETURN
1630 ' Codifica informação
1640 PP=0:QQ=0
1650 PP=(FIX(YR/100)-17)*16+MO
1660 K2=100:QQ=FNM(YR)
1670 KB$=CHR$(DA)+CHR$(PP)+CHR$
(QQ)+LEFT$(KB$,20):RETURN
1680 ' verifica tipo
1690 IF T7=0 THEN PRINT"Mensal,
 Trimestral, Anual, Unico":KB$=
"MTAU":GOSUB 1590:T7=KB:GOTO 17
20
1700 IF T7=2 THEN T7=3:GOTO 172
0
1710 T7=4
1720 RETURN
1730 ' gravar dados
1740 N=0:P=0
1750 OPEN "DIARIO" FOR OUTPUT A
S #1
1760 FOR N=0 TO 3
1770 M=VAL(LI$(N,0))
1780 PRINT#1,LI$(N,0)
1790 IF M=0 THEN 1840
1800 FOR P=1 TO M
1810 FOR J=1 TO 4:PRINT#1,STR$(
ASC(MID$(LI$(N,P),J,1))):NEXT
1820 PRINT#1,MID$(LI$(N,P),5)
1830 NEXT
1840 NEXT
1850 CLOSE#1
1860 RETURN
1870 ' carregar dados
1880 N=0:P=0:M=0
1890 OPEN "DIARIO" FOR INPUT AS
 #1
1900 FOR N=0 TO 3
1910 LINE INPUT#1,LI$(N,0)
1920 M=VAL(LI$(N,0))
1930 IF M=0 THEN 1980
1940 FOR P=1 TO M
1950 FOR J=1 TO 4:INPUT#1,NN$:L
I$(N,P)=LI$(N,P)+CHR$(VAL(NN$))
:NEXT
1960 LINE INPUT#1,NN$:LI$(N,P)=
LI$(N,P)+NN$
1970 NEXT
1980 NEXT
1990 CLOSE#1
2000 RETURN
2010 ' calendário anual
2020 M4=0:A4=0
2030 INPUT"Ano: ";YR:IF YR<1753
 OR YR>29999 THEN 2030 ELSE GOS
UB 650
2040 GOSUB 2720:CLS
2050 PRINT"Ano ";YR
2060 IF P=2 THEN LPRINT "Ano: "
;YR:LPRINT
2070 PRINT:KB=0:GOSUB 2150:PRIN
T:IF P=2 THEN LPRINT
2080 GOSUB 2660
2090 FOR MO=1 TO 12
2100 PRINTMID$(MN$,MO*9-8,9):IF
 P=2 THEN LPRINT MID$(MN$,MO*9-
8,9)
2110 T2=5:S2=0:GOSUB 2240
2120 IF P=0 AND INKEY$="" THEN
2120
2130 NEXT
2140 RETURN
2150 ' imprime dias
2160 X2=0:C2=0:D2=0
2170 IF KB=0 THEN X2=7
2180 IF P=2 THEN KB=KB+1
2190 PRINT#P,STRING$(X2,32);
2200 FOR D2=0 TO 6
2210 PRINT#P, STRING$(KB," ")+M
ID$(DN$,D2*3+1,3);
2220 NEXT
2230 RETURN
```



```
115 KB$ = "SN": GOSUB 1590: IF
KB = 1 THEN  GOSUB 1870
1180  REM  LISTAR E ATUALIZAR
1190 N = 0:A = 0:B = 0:KK = KB
1200 KB = KK: GOSUB 1330: POKE
34,1: POKE 35,22
1210  REM
1220  HOME
1230 B =  VAL (LI$(KK,0))
1240  IF B = 0 THEN 1280
1250  FOR N = 1 TO B
1260 KB$ = LI$(KK,N):K2 = N: GO
SUB 1410
1270  NEXT
1280 KB$ = "ADM": GOSUB 1590:A
= KB
1290  IF A = 1 THEN T7 = KK: GO
SUB 1680: GOSUB 1490:LI$(KK,B +
 1) =  CHR$ (T7) + T8$:LI$(KK,0
) =  STR$ (B + 1): GOTO 1310
1300  IF A = 2 AND B > 0 THEN
INPUT "QUAL NUMERO? ";NN: IF NN
 < 1 OR NN > B THEN 1300
1305  IF A = 2 AND B > 0 THEN
FOR PP = NN + 1 TO B:LI$(KK,PP
- 1) = LI$(KK,PP): NEXT :LI$(KK
,0) =  STR$ (B - 1)
1310  IF A <  > 3 THEN 1210
1320  POKE 34,0: POKE 35,24: RE
TURN
```

```
1330  REM  CABECALHO
1340  PRINT TYS(KB),"LISTA CORR
ENTE"
1350  REM
1360  CALL  - 958
1370  REM
1380  VTAB 23: HTAB 5: PRINT "[
A]DICIONAR  [D]ELETAR  [M]ENU"
1390  REM
1400  RETURN
1410  REM  OP
1420 N2 = 0:BB$ = "":DD$ = "":K
3 = K2
1430  FOR N2 = 1 TO 4: IF  MID$
 (KB$,N2,1) = "" THEN CO(N2) =
0: GOTO 1440
1435 CO(N2) =  ASC ( MID$ (KB$,
N2,1))
1440  NEXT
1450 BB$ =  MID$ (PA$,CO(1) * 6
 - 5,6)
1460 K2 = 16:DD$ =  STR$ (CO(2)
) + ":" +  STR$ ( FN M(CO(3)))
+ ":" +  STR$ (( INT ( ABS (CO(
3) / 16)) *  SGN (CO(3) / 16) +
 17) * 100 + CO(4))
1470  PRINT  STR$ (K3);" ";BB$;
" ";DD$;" "; RIGHT$ (KB$, LEN (
KB$) - 4)
1480  RETURN
1490  REM  ADICIONAR DADOS
1500 B3$ = "":VP = 0
1510  PRINT "TITULO? (MAX 22 LE
TRAS)": INPUT B3$
1520  REM
1530  PRINT
1540  PRINT "DATA DE REFERENCIA
?"
1550  INPUT " DIA:";DA: IF DA <
 1 OR DA > 31 THEN 1530
1560  GOSUB 2750
1570 KB = MO: GOSUB 230: IF DA
> KB THEN 1530
1580 KB$ = B3$: GOSUB 1630:T8$
= KB$: RETURN
1590  REM  VERIF TECLAGEM DE CA
RACT EM KB$ E DEVOLVE POSICAO E
M KB
1600  GET B$:KB = 0
1610  FOR I = 1 TO  LEN (KB$):
IF  MID$ (KB$,I,1) = B$ THEN KB
 = I
1615  NEXT : IF KB = 0 THEN 160
0
1620  RETURN
1630  REM  CODIFICAR INFORMACAO

1640 PP = 0:QQ = 0
1650 PP = ( INT (YR / 100) - 17
) * 16 + MO
1660 K2 = 100:QQ =  FN M(YR)
1670 KB$ =  CHR$ (DA) +  CHR$ (
PP) +  CHR$ (QQ) +  LEFT$ (KB$,
22): RETURN
1680  REM  INDICA TIPO
1690  IF T7 = 0 THEN  PRINT "ME
NSAL, TRIMESTRAL, ANUAL, UNICO"
:KB$ = "MTAU": GOSUB 1590:T7 =
KB: GOTO 1720
1700  IF T7 = 2 THEN T7 = 3: GO
TO 1720
1710 T7 = 4
1720  RETURN
```




```
1730  REM  GRAVAR MATRIZ
1740 N = 0:P = 0
1750  PRINT D$;"OPEN DIARIO"
1755  PRINT D$;"WRITE DIARIO"
1760  FOR N = 0 TO 3
1770 M =  VAL (LI$(N,0))
1780  PRINT M
1790  IF M = 0 THEN 1840
1800  FOR P = 1 TO M
1810  FOR J = 1 TO 4: PRINT  ST
R$ ( ASC ( MID$ (LI$(N,P),J,1))
): NEXT
1820  PRINT  MID$ (LI$(N,P),5)
1830  NEXT
1840  NEXT
1850  PRINT D$;"CLOSE"
1860  RETURN
1870  REM  CARREGAR DADOS
1880 N = 0:P = 0:M = 0
1890  PRINT D$;"OPEN DIARIO"
1895  PRINT D$;"READ DIARIO"
1900  FOR N = 0 TO 3
1910  INPUT LI$(N,0)
1920 M =  VAL (LI$(N,0))
1930  IF M = 0 THEN 1980
1940  FOR P = 1 TO M
1950  FOR J = 1 TO 4: INPUT NN$
:LI$(N,P) = LI$(N,P) +  CHR$ (
VAL (NN$)): NEXT
1960  INPUT NN$:LI$(N,P) = LI$(
N,P) + NN$
1970  NEXT
1980  NEXT
1990  PRINT D$;"CLOSE"
2000  RETURN
2010  REM  CALENDARIO ANUAL
2020 M4 = 0:A4 = 0
2030  INPUT "ANO ",YR: IF YR <
1753 OR YR > 29999 THEN 2030
2040  GOSUB 650: GOSUB 2720: HO
ME
2050  PRINT "ANO ",YR
2060  IF P = 1 THEN PREPARAIMPR
ESS
2070 KB = 0: GOSUB 2150
2080  GOSUB 2660
2090  FOR MO = 1 TO 12
2100  PRINT  MID$ (MN$,MO * 9 -
 8,9)
2110 T2 = 5:S2 = 0: GOSUB 2240
2120  IF P = 0 THEN  GET R$
2130  NEXT
2140  RETURN
2150  REM  IMPRIME DIAS
2160 X2 = 0:C2 = 0:D2 = 0
2170  IF KB = 0 THEN X2 = 7
2180  IF P = 2 THEN KB = KB + 1

2190  PRINT  SPC( X2)
2200  FOR D2 = 0 TO 6
2210  PRINT  SPC( KB); MID$ (DN
$,D2 * 3 + 1,3);
2220  NEXT
2230  RETURN
```

PALETA ELETRÔNICA PARA O TK-2000

A imaginação ao poder! Siga o lema das barricadas parisienses de 1968, e transforme seu vídeo em uma tela de pintura, executando o programa deste artigo no velho e fiel TK-2000.

Os programas elaboradores de gráficos constituem uma das ferramentas mais poderosas e versáteis para os usuários de microcomputadores. É preciso notar, entretanto, que existem diversos tipos de programas gráficos. Os mais comuns permitem traçar gráficos de análise de dados, como histogramas, curvas, segmentos circulares ("pizzas") etc., e têm muitas aplicações em educação, ciências e negócios.

Outro tipo de muita aceitação consiste nos programas "desenhistas", que fazem desenhos livremente sobre a tela, usando o teclado ou um dispositivo de entrada gráfica ("*mouse*" ou *joystick*) para deslocar um cursor gráfico sobre o vídeo. Esses programas, disponíveis para a maioria dos microcomputadores, principalmente para os que têm boa capacidade gráfica e de impressão em cores (Spectrum, Apple, MSX, IBM, PC etc.), contam com um grande número de recursos, tais como: figuras geométricas pré-programadas (círculos, elipses, retângulos etc.); "pincéis" de várias espessuras; cores para desenhar e pintar; dispositivos para "cortar e colar" (retirar uma parte de um desenho e colocar em outro ponto da tela) e "carimbar" (copiar uma parte de um desenho em outro ponto da tela); recursos para escrever títulos e textos com letras de diversos estilos e tamanhos etc. Programas como esses constituem uma verdadeira "paleta eletrônica". Alguns deles são bem conhecidos; é o caso de *Paint*, *Paint-Brush*, *Dr. Halo*, *Apple-Paint* etc.

O programa apresentado aqui pertence a essa categoria, embora seja muito mais simples. Ele permite a elaboração de letreiros e gráficos, empregando o conceito de cursor gráfico. Os recursos de programação de que ele dispõe não são muito complexos, e consistirão num bom aprendizado para quem quiser conhecer os "truques" básicos usados em programas do mesmo tipo.

COMANDOS SIMPLES

A filosofia de operação do programa é simples: ao ser inicializado, o programa entra em modo gráfico de baixa resolução (GR) e coloca um cursor branco no canto superior esquerdo da tela. Ao pressionar as teclas correspondentes a cada comando, o usuário pode deslocar o cursor sobre a tela, trocar sua cor, desenhar com a cor selecionada, apagar traços, desenhar blocos retangulares (preenchidos com cor) ou molduras (retângulos vazios) etc. Existem também comandos para escrever um texto em qualquer ponto da tela, em vídeo direto ou vídeo inverso.

Compõe-se, assim, uma ilustração que depois pode ser fotografada diretamente da tela — ou copiada —, em preto e branco, em uma impressora gráfica. Portanto, além de ser uma diversão para quem quer apenas exercitar os seus dotes criativos na pintura, o programa é muito bom para a produção de dispositivos coloridos destinados a ilustrar palestras, aulas, apresentações etc.

O cursor pode se movido pela tela, usando-se as teclas ↑, ↓, → e ←. Para mudar sua cor, pressiona-se a tecla **C**, e logo em seguida digita-se um dos números compreendidos entre 0 e 6 (cada um deles correspondendo a uma das cores disponíveis).

Para traçar uma reta com a cor do cursor, pressiona-se a tecla **<FIRE>** e movimenta-se o cursor na direção desejada. Para parar de traçar, aciona-se **<FIRE>** novamente. Assim, esse sistema de operação serve ainda para apagar trechos de um desenho: basta selecionar a mesma cor do fundo para o cursor, e deslocá-lo sobre a área que deve ser apagada. No TK 2000, o joystick pode ser usado para movimentar o cursor e também para traçar retas, pois seu acionamento corresponde às teclas do cursor e à tecla **<FIRE>**.

O programa tem oito comandos apenas (no final deste artigo vamos dar algumas "dicas" sobre como expandir o programa, adicionando outros tipos de comandos). Eis aqui os citados comandos, em ordem alfabética:

- **B** - traça uma borda na tela, na cor previamente indicada.
- **C** - muda a cor do cursor. Um número de 0 a 6 será, então, digitado.
- **F** - preenche toda a tela com a cor designada pelo cursor.
- **I** - passa a aceitar um texto qualquer digitado pelo teclado, escrevendo-o em vídeo inverso (letras escuras sobre fundo claro). O cursor gráfico não se desloca, permanecendo na posição inicial. Pressionando-se a tecla **<RETURN>**, o sistema volta ao modo gráfico.
- **R** - retorna o cursor à sua posição de partida (canto superior esquerdo da tela) e limpa o vídeo.
- **M** - traça uma moldura (retângulo vazio) com a cor e posição indicadas pelo cursor (o canto superior esquerdo do retângulo ficará posicionado na localização do cursor). O programa solicita o comprimento e a largura do retângulo (em pixels), que devem ser digitados separados por uma vírgula.
- **Q** - traça um quadro (retângulo cheio), com a cor e posição indicadas pelo cursor. O restante funciona como no caso do comando **M**.
- **T** - passa a aceitar um texto qualquer digitado pelo teclado, escrevendo-o em vídeo direto ou normal (ou seja, letras claras sobre fundo escuro). O restante funciona do mesmo modo que no comando I.

Depois que um texto for colocado em algum ponto da tela, deve-se evitar passar com o cursor sobre ele; do contrário, ele será apagado.

COMO FUNCIONA O PROGRAMA

O programa é dividido em quatro grandes seções de processamento:

A seção que vai da linha 50 até a linha 60 inicializa todos os parâmetros usados no programa e prepara ao mesmo tempo a tela inicial.

A seção que vai da linha 100 à 175 aceita comandos através do teclado e verifica qual foi a tecla pressionada. Se for um dos comandos válidos, ela modificará os parâmetros internos do programa, ou chamará a rotina encarregada de executar a função desejada.

A seção que vai da linha 180 à 200 serve para testar se os limites da tela foram ultrapassados pelo movimento do cursor, e traça no vídeo um quadradi-







nho na cor vigente, na posição indicada (coordenadas do cursor).

Finalmente, as linhas 500 a 900 contêm diversas sub-rotinas de trabalho, que explicaremos mais adiante.

Comecemos com a primeira seção (não a execute ainda; caso contrário, ocorrerá uma mensagem de erro):

```
50 GR· COLOR = 0
55 LET FL = 0: X = 0: Y = 0
56 LET QX = 39: QY = 39: GOSUB
   815
60 LET XL = 1: YL = 1: CL = 0:
   C = 3: GOSUB 196
```

A linha 50 estabelece o modo gráfico de baixa resolução (GR) bem como a cor inicial (preto) do cursor. A linha 55 inicializa as variáveis **FL** (um indicador lógico, que mostra se o modo de desenhar está "ligado" ou "desligado"), **X** e **Y** (coordenadas correntes do cursor; note que **X** = **0** e **Y** = **0** correspondem, no TK-2000, à posição no canto superior esquerdo).

A linha 56 define, por sua vez, as variáveis **QX** e **QY**, necessárias para a atuação da rotina da linha 815 (mostrada mais adiante num dos segmentos do programa), que é responsável por traçar um bloco colorido. Quando as variáveis **QX** e **QY** são igualadas a 39, como neste caso, isso significa que a tela será inteiramente pintada com a cor corrente do cursor (na primeira vez será preto, ou **COLOR** = **0**).

A linha 60 define as variáveis **XL** e **YL**, que servem para armazenar as últimas coordenadas do cursor, atingidas antes das correntes (**X** e **Y**), bem como a última cor **CL**. Ainda nessa linha, é definida a cor inicial do cursor (**C** = **3**), e a sub-rotina da linha 196 é chamada para traçar esse cursor em sua posição inicial (0,0).

Digite agora a seção de entrada e teste (sem, contudo, apagar a seção de entrada anterior):

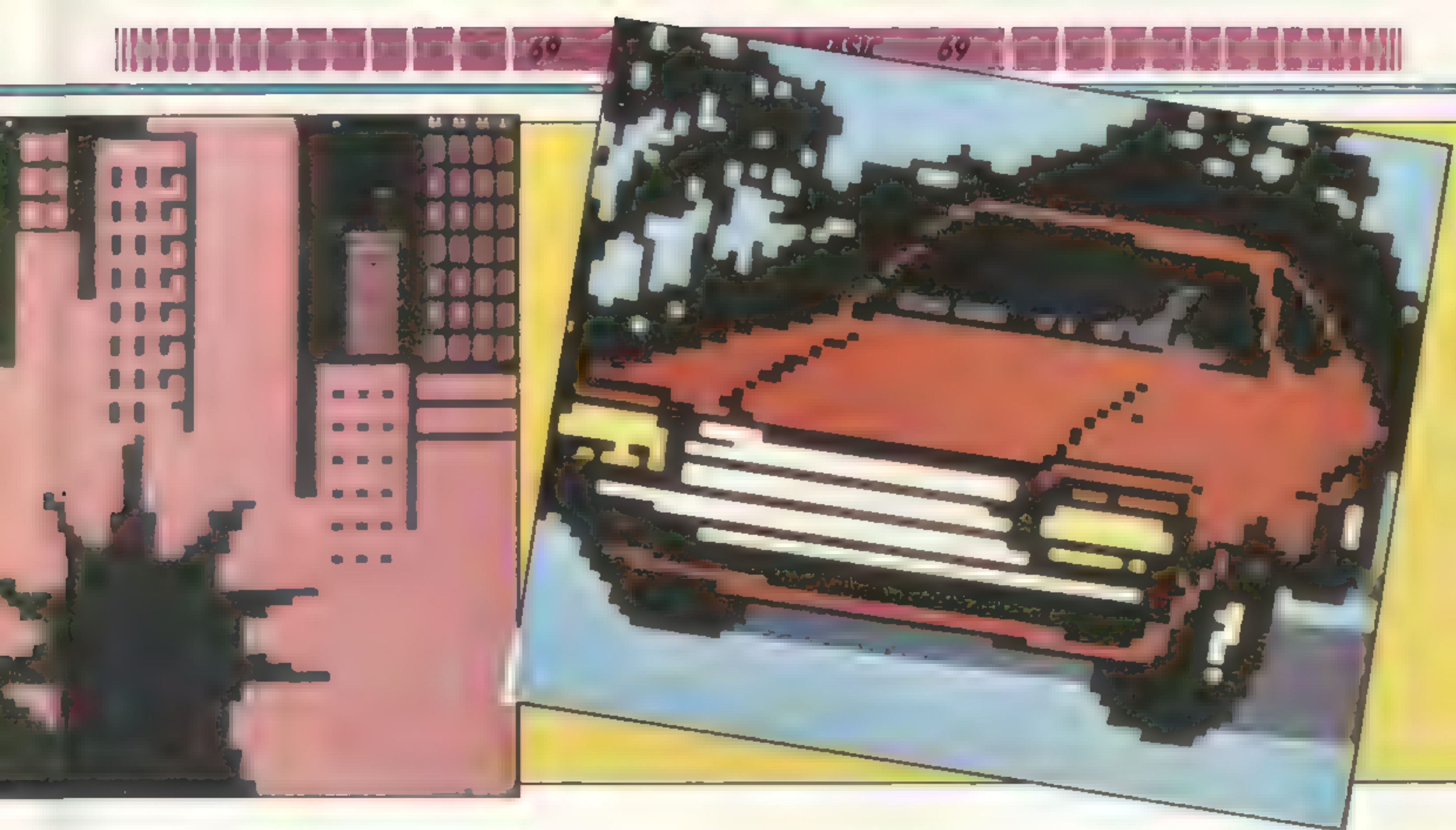
```
100 GET A$: A = ASC (A$)
105 IF A$="C" THEN GOSUB 850:
    GOTO 100
110 IF A$="Q" THEN GOSUB 600·
    GOTO 100
115 IF A$="M" THEN GOSUB 650:
    GOTO 100
120 IF A$="T" THEN GOSUB 500·
    GOTO 100
125 IF A$="I" THEN INVERSE·
    GOSUB 500: GOTO 100
130 IF A$="R" THEN HOME: GOTO
    55
140 IF A$="B" THEN COLOR = C:TY
    = 1: X = 0: Y = 0: GOSUB
    815: X = XL: Y = YL: GOTO
    100
145 IF A$="F" THEN COLOR = C:TY
    = 0· X = 0: Y = 0: GOSUB
    815: X = XL: Y = YL: GOTO
    100
150 IF A = 112 THEN Y = Y-1:
    GOTO 190
155 IF A = 113 THEN Y = Y+1:
    GOTO 190
160 IF A = 8 THEN X = X-1: GOTO
    180
165 IF A = 21 THEN X = X+1: GO-
    TO 180
166 IF A <> 46 THEN GOTO 100
170 IF FL = 1 THEN FL = 0: GOTO
    100
172 IF FL = 0 THEN FL = 1
175 GOTO 100
```

A linha 100 usa um comando **GET** para permitir uma pressão à tecla e armazenar o caractere correspondente em **A$**. Ao mesmo tempo, o valor ASCII desse caractere é determinado pela função **ASC** e guardado em **A**.

Em seguida, as linhas 105 a 145 verificam se o caractere digitado é **C**, **Q**, **M**, **T**, **B**, **R**, **I** ou **F**. Em caso positivo, elas tomam as providências necessárias para executar a função perdida, e retornam à linha 100 para receber mais um comando.

Essa estrutura de programa caracteriza, de fato, um laço sem fim (conhecido, neste caso, como *laço de entrada*), que não chega a ser um laço de espera verdadeiro, pois o programa ficará parado na linha 100 enquanto nenhuma tecla for pressionada.

As linhas entre 150 e 175 servem para verificar se uma das teclas de movi-



mentação do cursor foi pressionada. Aqui, temos que usar o valor ASCII para fazer a averiguação, pois não existe representação gráfica da tela para os cursores. Se a tecla tiver sido pressionada, o parâmetro **Y** (coordenada vertical do cursor) será diminuído de uma posição, e o programa saltará, então, para a linha 190, que verifica se os limites da tela foram ultrapassados. Transformações semelhantes de **X** e **Y** são feitas, conforme a tecla de cursor seja pressionada. Mas não execute o programa ainda.

COMECE A DESENHAR

Adicione as linhas abaixo:

```
180 IF X < 0 THEN X = 0
182 IF X > 39 THEN X = 39
185 GOTO 196
190 IF Y < 0 THEN Y = 0
195 IF Y > 39 THEN Y = 39
196 IF FL = 1 THEN GOTO 198
197 COLOR = CL: PLOT XL,YL
198 LET CL = SCRN (X,Y): COLOR
    = C: PLOT X,Y
199 LET XL = X: YL = Y
200 GOTO 100
```

As linhas 180 a 185 avaliam se **X** ultrapassou ou não a borda da tela, mantendo-o nesse limite, caso isso tenha acontecido. As linhas 190 e 195 fazem o mesmo com **Y**.

A linha 196 verifica se o cursor está em modo de desenho. Se estiver, a linha 197 traçará o cursor, com a cor **CL** (última cor encontrada), na posição (**XL**, **YL**). O programa prosseguirá então para a linha 198, que testa a cor do pixel em **X**, **Y**, armazena-o em **CL**, e traça finalmente o pixel em **X** e **Y**, usando a cor **C**.

Se esse procedimento, aparentemente complexo, não for aplicado, toda vez que o cursor for deslocado deixará um "buraco" de cor diferente na posição anterior.

Por fim, nas linhas 199 e 200, são usados os valores das variáveis **XL** e **YL**, e o programa retorna à linha 100, ponto de partida desse segmento, para mais uma "rodada" de entrada.

Uma vez concluídos esses preparativos, podemos adicionar as rotinas de trabalho do programa:

```
500 LET YT = Y/2: XT = X+1
510 HTAB XT: VTAB YT
515 GET A$: IF ASC (A$) = 13
    THEN NORMAL: RETURN
516 IF ASC (A$) = 8 THEN XT =
    XT 1: GOTO 510
520 PRINT A$: XT = XT+1
525 GOTO 510
600 LET TY = 0: GOTO 750
650 LET TY = 1
750 HTAB 1: VTAB 23
800 INPUT "COMPRIMENTO,LARGURA
    :";QX,QY
815 IF TY = 1 THEN GOTO 840
820 FOR I = Y TO Y+QY
830 HLIN X,X+QX AT I
835 NEXT I: CL = C: GOTO 900
840 HLIN X,X+QX AT Y: HLIN
    X,X+QX AT Y,Y+QY
847 LET CL = C: GOTO 900
850 HTAB 1: VTAB 23: INPUT "COR
    (0-60):";C
855 IF C<0 OR C>7 THEN GOTO 850
860 COLOR = C: PLOT X,Y
900 HTAB 1: VTAB 23: PRINT SPC
    (39): RETURN
999 END
```

A sub-rotina que vai da linha 500 à linha 525 responde aos comandos **I** e **T** e serve para receber caracteres pelo teclado (linha 515) e colocá-los na tela, a partir da última posição do cursor de texto.

As variáveis **XT** e **YT** correspondem às posições de tabulamento de texto, nas posições **HTAB** e **VTAB**, que são calculadas a partir da localização do cursor gráfico no momento (**X** e **Y**). O caractere é impresso pelo comando **PRINT** na linha 520, onde também é incrementado o cursor de texto (**XT**).

Ao mesmo tempo, a entrada de um comando <**RETURN**> (linha 515, com código 13) encerra a sub-rotina e volta para a alça de espera. O código

8, por sua vez, corresponde ao comando **BACKSPACE** no micro TK-2000 e faz com que o cursor recue uma posição, apagando o caractere escrito.

A sub-rotina que vai da linha 600 a linha 847 e um procedimento universal para traçar retângulos, sejam eles coloridos ou não. Ela serve, portanto, para executar os comandos **B**, **R**, **Q** e **M** (os parâmetros adequados para o caso são fixados nos **IF** das linhas 105 a 145).

A variável **TY** serve para assinalar se o retângulo é cheio ou vazio. A linha 800 pergunta as suas dimensões, no caso dos comandos **Q** e **M**, e à linha 810 cabe investigar se ele não excede os limites da tela.

Finalmente, a rotina que vai de 850 a 900 muda a cor do cursor. Na linha 900, é apagada a última linha de texto da tela, de modo a prepará-la para o próximo comando.

INCREMENTANDO...

Execute agora, com todo o cuidado, o programa completo e dê asas à sua imaginação. Entretanto, não esqueça de gravá-lo antes em fita.

Com um pouco de criatividade e algum esforço de programação, não é difícil aumentar a utilidade do programa, adicionando-lhe novos comandos. Para isso, basta inserir uma nova linha **IF** na série de linhas desse tipo existente entre as linhas 100 e 165, verificando se a nova tecla foi pressionada ou não, e desviar o processamento com um comando **GOSUB** para a sub-rotina que fará a tarefa.

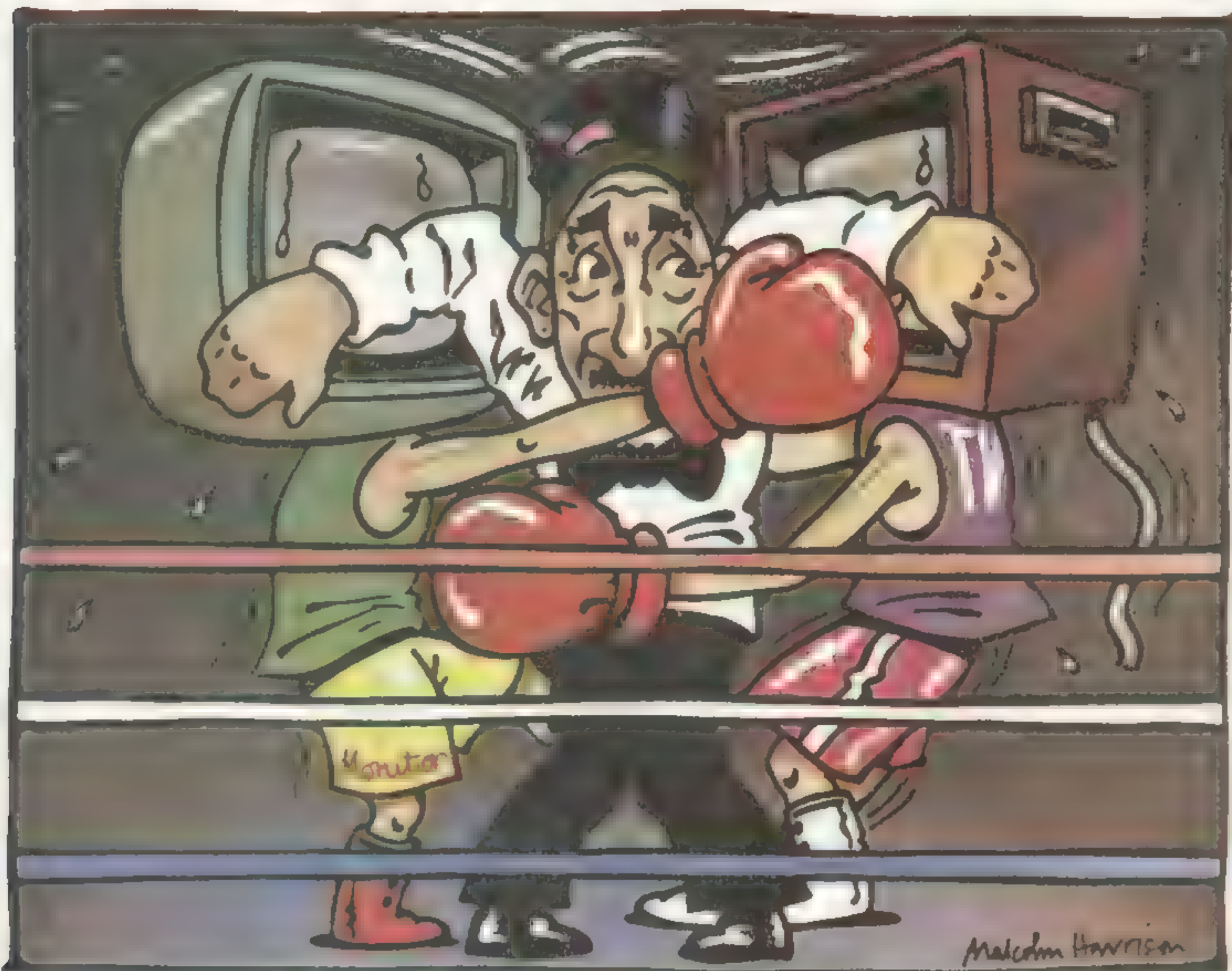
Eis aqui algumas sugestões de extensão do programa:

- sub-rotina para traçar retas de uma só vez (marque os pontos inicial e final da reta, com o cursor, e pressione **<ENTER>** para acionar a rotina).
- sub-rotina para copiar blocos de um lugar para outro.
- sub-rotina para apagar um bloco retangular inteiro da tela, reconstituindo a cor de fundo.
- rotinas para gravar telas em fita e carregá-las de volta à tela (leia a tela gráfica com comandos **POKE**).
- rotinas para criar uma biblioteca de "ícones" (pequenos desenhos ou motivos gráficos), que podem ser copiados em qualquer ponto da tela.



TV VERSUS MONITORES

- MONITORES E TELEVISORES
- SINAIS DE ENTRADA
- COR VERSUS PRETO E BRANCO
- DEFINIÇÃO DA IMAGEM
- COMO ESCOLHER



Monitores ou televisores? Se você tem dúvidas a respeito do vídeo ideal para o seu computador, leia este artigo, onde são abordadas as diferenças entre os dois aparelhos.

Componente essencial para a operação dos microcomputadores, a tela de vídeo constitui o principal canal de comunicação entre o usuário e a máquina. Além disso, é o vídeo que mantém visível o registro de tudo o que é digitado no teclado. Sofisticado e sensível, esse equipamento eletrônico deve ser bem selecionado no momento da compra, pois o seu desempenho influencia diretamente o da máquina.

Existem basicamente duas opções de saída de vídeo para os microcomputadores: o uso direto de um aparelho de TV (em cores ou em preto e branco), ou um monitor de vídeo (que também pode ser monocromático ou colorido). Muitas máquinas já vêm equipadas com um monitor de vídeo padrão, que dificilmente pode ser trocado. Se o seu computador é desse tipo, nada resta a fazer. Caso contrário, vale a pena ler as linhas a seguir. Elas explicam como funciona um desses equipamentos e como o computador controla a saída de vídeo.

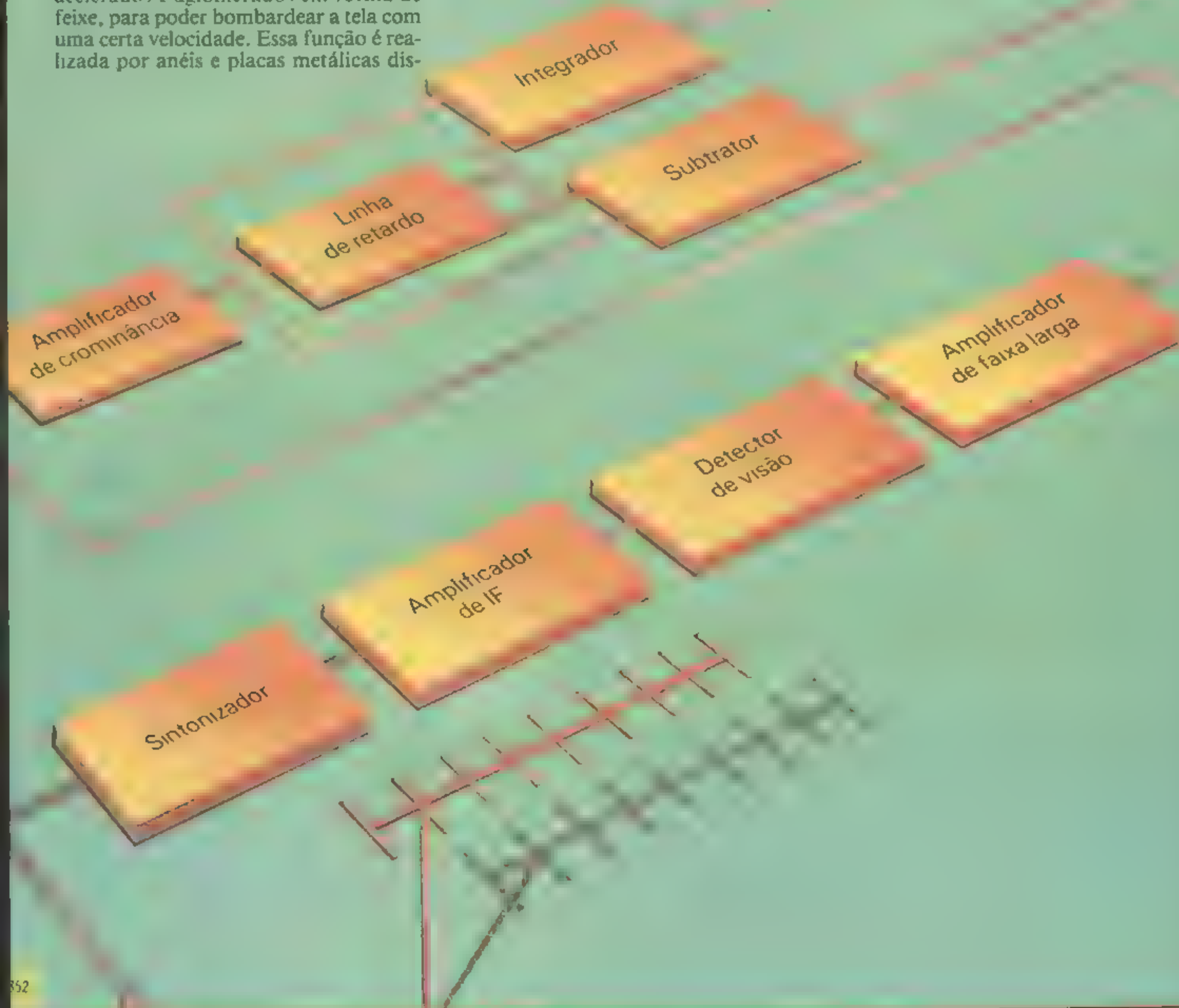
Embora existam muitas maneiras diferentes de passar informações do com-

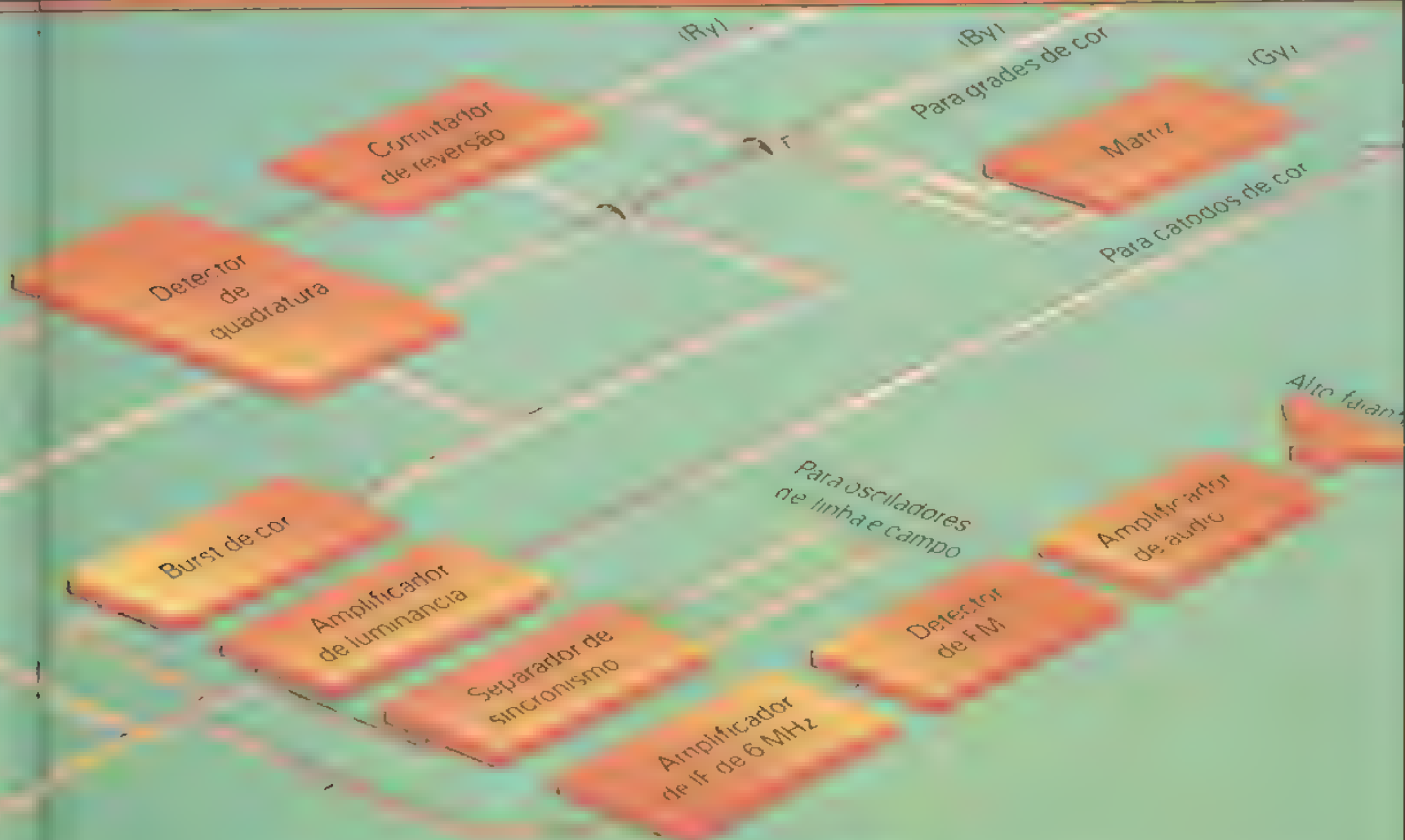
putador para a tela, os diversos tipos de vídeo contam com um dispositivo especialmente destinado para isso: é o *tubo de raios catódicos*, conhecido por sua sigla em inglês, CRT (*Cathode-Ray Tube*). Os tubos de raios catódicos (chamados também de cinescópios) funcionam com base no fenômeno da fosforescência, ou luminosidade provocada por feixes de elétrons em certas substâncias químicas, como fósforo.

Um tubo de raios catódicos consiste em um invólucro de vidro em forma de cone ou pirâmide, em cujo interior se produz vácuo. Em uma de suas extremidades é montado um canhão de elétrons (um filamento aquecido, emissor de elétrons). Uma vez liberados, os elétrons compõem uma espécie de nuvens ao redor do filamento; eles são então acelerados e aglomerados em forma de feixe, para poder bombardear a tela com uma certa velocidade. Essa função é realizada por anéis e placas metálicas dispostas ao longo do tubo, que são eletrificadas — positivamente, para acelerar os elétrons, e negativamente, para obrigá-los a constituir um fino feixe, que, ao atingir a tela, produzirá um ponto de luz.

Para criar imagens na tela, é necessário fazer esse ponto de luz se deslocar em várias direções sobre a camada de fósforo. Isso se dá através de dois pares de placas, dispostas em ângulo reto: são as *placas defletoras*, que realizam sua função por meio de voltagens elétricas; estas, por sua vez, atraem ou repelem o feixe nas direções horizontal e vertical.

Os aparelhos de TV e a maioria dos monitores de vídeo criam imagens e textos empregando a técnica de *varredura*.





Nesta, um sinal eletrônico especial é aplicado às placas defletoras, de modo que o ponto de luz percorre todo o vídeo a grande velocidade, em um movimento de ziguezague. Isso permite formar na tela entre quinhentas e 625 linhas (dependendo, evidentemente, do padrão adotado), em cerca de 1/60 de segundo. Esse padrão de varredura é constantemente repetido.

Para produzir uma imagem ou texto, portanto, basta *modular* a intensidade do feixe, ou seja, variar a voltagem do canhão de elétrons, de modo a ligá-lo e desligá-lo em determinados momentos. Assim, por exemplo, os caracteres podem ser criados por meio de pontinhos acesos. Portanto, tudo o que o controlador de vídeo do computador precisa fazer é enviar esse sinal de intensidade ao CRT.

Alguns tipos de monitores especiais para micro formam imagens ou caracteres utilizando a modulação das placas defletoras. Embora essa técnica produza imagens de excelente definição e qualidade, uma boa velocidade de exibição somente pode ser obtida a custos muito altos. Na verdade, o CRT de varredura gera imagens complexas de maneira mais barata e rápida.

PERSISTÊNCIA VISUAL

O uso da técnica de varredura para gerar imagens é baseado em uma propriedade da visão humana, que, em outras circunstâncias, poderia ser considerada um defeito. O olho humano retém uma imagem na retina por cerca de 1/25 de segundo. Como a maioria dos tubos de raios CRT é capaz de varrer toda a tela em cerca de 1/60 de segundo, isto significa que uma nova imagem, ou "quadro", aparece mais rapidamente do que o olho pode perceber. Assim, vemos na tela uma imagem que se mantém, ou se move, sem piscar. Se nossos olhos tivessem uma capacidade de retenção de 1/100 de segundo, ou menos, a imagem gerada em um televisor pareceria piscar intoleravelmente.

COLORIDO

Esse princípio vale tanto para televisores em preto e branco (monocromáticos), quanto para os que transmitem imagens coloridas. A única diferença entre os CRT dos dois tipos é que os aparelhos monocromáticos têm apenas um canhão eletrônico e uma variedade de fósforo na tela, enquanto os equipamentos coloridos dispõem normalmente de três canhões e três espécies de fósforo (existe ainda um tipo de CRT colorido que possui apenas um canhão que dispara três feixes de elétrons).

A cor é produzida da seguinte maneira: os feixes eletrônicos ativam os fósforos, que se iluminam, assumindo, cada um, uma cor diferente (azul, vermelho ou verde).

SINAIS DE ENTRADA

Monitor ou televisor, monocromático ou em cores, o vídeo é preparado para aceitar três tipos de sinal de entrada. O primeiro deles, o sinal padrão de radioemissão, é usado por aparelhos de televisão; ele produz a imagem convencional de TV na tela e é conhecido por RF (abreviatura de *radiofrequência*, usualmente na faixa VHF).

Chamado de *vídeo composto*, o segundo tipo de sinal é utilizado pela maioria dos monitores, embora alguns aparelhos de TV também tenham uma entrada de vídeo composto já embutida. Esses sinais contêm a modulação de

intensidade para os feixes eletrônicos, além de pulsos de sincronização.

Finalmente, existe o *sinal RGB*, cujo nome vem das iniciais das três cores: **R**ed, **G**reen, **B**lue (vermelho, verde e azul, em inglês). Ele constitui a forma mais direta e precisa de entrada de cores. Neste caso, cada canhão eletrônico é alimentado por informações acerca da cor correspondente (há, porém, monitores monocromáticos capazes de aceitar sinais RGB, transformando-os em níveis de intensidade, também chamados de níveis de cinza).

O sinal de vídeo composto é intermediário, em qualidade e flexibilidade, entre o RF e o RGB. O sinal RF, em última instância, passa por uma série de amplificações e transformações, antes de ser convertido em um sinal de vídeo direto. Portanto, ele é de menor qualidade e definição, pois, quanto maior for o número de estágios intermediários, maiores serão as distorções e interferências sofridas por ele.

LARGURA DE FAIXA

Existem razões interessantes, mas tecnicamente complicadas, para o fato de um televisor não possibilitar o mesmo grau de definição de imagem na tela que um monitor de vídeo. O elemento diferenciador é, aqui, a chamada largura de faixa (*bandwidth*, em inglês), que é a diferença algébrica entre as frequências máxima e mínima de sinal, às quais o dispositivo eletrônico e o CRT podem responder sem perda apreciável de intensidade (ou ganho). Do mesmo modo, a largura de faixa é o principal parâmetro a determinar a qualidade ou resolução de imagem.

Um aparelho de TV, por exemplo, não aceita sinais que requeiram uma largura de faixa maior do que 5,5 MHz (megahertz, ou 1 000 000 de oscilações por segundo). Essa faixa funciona bem em um vídeo de 32 ou quarenta colunas; neste último caso (quarenta colunas), porém, os caracteres tendem a tornar-se indefinidos ou "borrados" (é que essa dimensão de vídeo exige, normalmente, uma largura mínima de 10 MHz).

Os monitores de vídeo são projetados especificamente para processar faixas de tal magnitude. Entretanto, é necessário que o computador tenha uma saída de vídeo composto, em vez de uma RF (algumas vezes ambas estão presentes, como é o caso dos micros da linha MSX). Os modelos da linha Sinclair, por exemplo (ZX 81 e Spectrum), têm saídas somente para RF.

Quando falamos da qualidade da imagem na tela, estamos nos referindo à resolução (ou definição de imagem) e à separação de cores. Infelizmente, quase sempre as duas características são antagônicas, ou seja, a presença de muitas cores implica menor definição, pois a largura de faixa tem que ser dividida entre as frequências presentes nos três canhões. Assim, um monitor monocromático produz uma definição melhor do que os tubos cromáticos.

Em uma tela normal de vídeo, de qualquer espécie, existem cerca de 360 000 pontos de fósforo. No caso de uma tela em cores, 120 000 desses pontos serão vermelhos, 120 000, verdes, e 120 000, azuis. Portanto, a resolução da imagem será três vezes menor do que a de um tubo monocromático.

Para muitas aplicações, como o processamento de textos, o monitor monocromático é a opção natural, pois força menos os olhos do usuário. A combinação mais freqüente é verde sobre fundo preto. O vídeo inverso, ou seja, caracteres pretos sobre fundo branco ou de cor clara, é menos frequente, pois provoca maior cansaço visual.

É bom lembrar, aliás, que a maioria das pesquisas a respeito dos efeitos do vídeo sobre a saúde não conseguiram demonstrar a nocividade do CRT.

Os efeitos maléficos são geralmente causados por outros fatores, como brilho refletido na tela, cores inadequadas, caracteres imprecisos. Uma tela de TV, entretanto, pisca imperceptivelmente — embora mais que a de um monitor —, o que pode provocar dores de cabeça e problemas visuais.

É importante notar ainda que a definição de imagem não depende apenas do CRT e dos seus circuitos de controle, mas também do hardware do computador. Os caracteres são formados por "células" ou matrizes de pontinhos de 5 × 7, 7 × 9, ou 9 × 9, conforme a qualidade do micro. Combinada com a largura de faixa do monitor, essa configuração pode produzir caracteres de pequena ou de grande resolução.

Por outro lado, embora o ideal fosse associar a cada pontinho luminoso na tela um bit ou um conjunto de bits na memória do computador (no caso de informação em cores são necessários de três a sete bits), esta seria uma solução pouca prática e muito cara. Assim, normalmente, um ou mais bytes da RAM são associados a caracteres (matrizes de pontinhos) e não a pontinhos individuais, e o circuito gerador de vídeo do computador faz o resto do trabalho, utilizando uma tabela interna de caracteres (que pode ser alterada em alguns micros, como no Sinclair).

COMO ESCOLHER

Os dois fatores mais importantes a serem considerados quando se deseja comprar um vídeo são: o tipo de computador ao qual ele deve ser acoplado, e as aplicações a que ele está destinado. Embora a tela seja um elemento importante do sistema, ela depende do tipo de hardware de vídeo do computador. Se a capacidade gráfica do micro é pobre, não adianta comprar um monitor de altíssima resolução, que ela não vai melhorar: você estará jogando dinheiro fora. Portanto, é preciso comprar o vídeo mais adequado para aquilo que o computador tem a oferecer.

Ao mesmo tempo, é importante também levar em conta a função para a qual o computador está sendo destinado. Assim, o monitor monocromático é a solução mais adequada quando se procura uma melhor relação custo/benefício em aplicações nas quais a cor não é um pré-requisito. Se, ao contrário, é necessário um padrão de cor de alta qualidade, então um bom monitor RGB deve ser considerado, embora a maioria dos micros pessoais e domésticos não precise de nada mais sofisticado do que um monitor de vídeo composto.

Jogos e programações pouco complexas requerem apenas um aparelho de TV como saída de vídeo. Evidentemente, a maioria dos jogos aparece de forma muito melhor em um monitor em cores; entretanto, quase todos eles são programados em função das limitações dos televisores. Além disso, o monitor pode ser uma má escolha com relação a outros aspectos. Por exemplo, normalmente os monitores de vídeo não têm saída sonora, pois são fabricados para o mercado profissional. Ora, essa característica tem importantes implicações, pois, de modo geral, os jogos mais sofisticados e interessantes são programados com efeitos sonoros.

Também não é recomendável usar televisores baratos ou muito antigos. Em primeiro lugar, os circuitos eletrônicos (alguns ainda usando válvulas) podem não ser compatíveis com a saída analógica de vídeo dos computadores. Em segundo lugar, defeitos que não causam grande impacto sobre a qualidade da imagem de TV — como o "estouro" da varredura nas margens da tela, ou a acentuação de uma cor sobre as outras — afetam negativamente tanto o texto como as imagens geradas pelo computador. Neste caso, pode sair muito mais caro tentar reparar o velho aparelho com defeito do que comprar um monitor ou um televisor novos.

O BANDIDO DE UM BRAÇO SÓ

- UTILIZE TODA A CAPACIDADE GRÁFICA DO SEU MICRO
- AS ENGRENAGENS DA MÁQUINA
- MONTANDO O CAÇA-NÍQUEIS
- AS INSTRUÇÕES



Participe deste novo jogo de INPUT e viaje para o mundo mágico de Las Vegas, acionando a manivela de uma máquina caça-níqueis que não vai deixá-lo de bolsos vazios.

Atualmente, já é possível substituir os velhos caça-níqueis mecânicos por programas de microcomputadores, onde a tela é usada para mostrar as rodas com os desenhos de frutas. É exatamente isso que faremos neste e nos próximos artigos de *INPUT*.

O jogo desempenha as funções características de um caça-níqueis real — ou seja, prender, apostar e empurrar. As rodas móveis com os desenhos de frutas são representadas por meio de gráficos animados.

Como se não bastasse, ele tem a vantagem de poupar o jogador de perder grandes quantias em dinheiro para esses bandidos de um braço só — como são conhecidos os caça-níqueis em sua terra natal, os Estados Unidos. Em compensação — já que ninguém é perfeito —, o jogador não ganha nenhum tostão. Para começar, consulte a seção dedicada a seu micro.

Não se esqueça, entretanto, de gravar a primeira parte do programa, apresentada neste artigo, antes de passar à execução da segunda parte. O programa não funciona neste estágio, embora alguns microcomputadores mostrem o aspecto básico da tela gráfica.

S

OS BLOCOS GRÁFICOS

As linhas 140 a 150 contêm os bytes responsáveis pelos blocos usados para desenhar as frutas do caça-níqueis. As instruções coloridas devem ser digitadas diretamente por meio de códigos de controle, de modo a economizar memória. Para escrever essas linhas, entre no "modo estendido" (**CAPS SHIFT + SYMBOL SHIFT**) e pressione a tecla relativa à cor desejada. A seguir entre no modo gráfico (**CAPS SHIFT** e **9**) e digite as letras da listagem.

```
10 LET HFLAG=0: POKE 23658,8:
RESTORE 20: GOSUB 860
20 DATA 14,31,31,31,31,15,3,1
,56,252,252,252,252,248,224,
192
30 DATA 49,42,51,42,50,255,
255,255,152,84,216,84,84,255,
255,255
40 DATA 3,4,8,28,62,62,62,28,
28,190,125,62,28,0,0,0
50 DATA 0,0,16,28,15,7,3,0,4,
12,26,56,248,240,224,0
60 DATA 0,0,7,15,31,31,15,7,8
,248,240,240,224,224,192,128
70 DATA 1,3,3,7,15,15,24,1,
128,192,192,224,240,240,24,
128
80 DATA 1,7,15,31,31,31,31,15
,224,240,240,240,240,224,192,
128
90 DATA 8,8,8,8,73,42,28,8,16
,56,84,146,16,16,16,16
100 FOR i=USR "a" TO USR "p"+7
: READ a: POKE i,a: NEXT i
110 LET TOTAL=100
140 DATA "AB","AB","CD","CD","
IJ","AB","EF","KL","MN","EF","
","MN"
145 DATA "IJ","MN","KL","MN","
","","CD","CD","EF","AB","
CD","MN"
150 DATA "EF","MN","  ","KL","
CD","AB","IJ","AB","IJ","CD","
IJ","IJ"
160 DIM a$(24,4): DIM b$(24,4)
: DIM c$(24,4): FOR i=1 TO 12:
READ a$(i): LET a$(i+12)=a$(i)
: NEXT i
180 FOR i=1 TO 12: READ b$(i):
LET b$(i+12)=b$(i): NEXT i
190 FOR i=1 TO 12: READ c$(i):
LET c$(i+12)=c$(i): NEXT i
860 BORDER 7: PAPER 9: INK 2:
CLS
870 PRINT ""
880 FOR i=0 TO 1: FOR j=0 TO
31: PRINT PAPER 7;AT i,j;" "
: NEXT j: NEXT i
890 FOR I=2 TO 5: PRINT AT I,0
; PAPER 6;"        ";AT I,28;"
    ": NEXT I
900 PRINT PAPER 6; INK 2;AT 0
,6;"10 CENTAVOS A JOGADA"
910 PRINT PAPER 6; INK 2;AT 1
,4;"
        ";AT 2,4;"

920 PRINT PAPER 6; INK 2;AT 1
,4;"
        ";AT 3,4;"
                        "
930 PRINT PAPER 6; INK 2;AT 1
,4;"
        ";AT 4,4;"
                        "
940 PRINT PAPER 6; INK 2;AT 1
,4;"
        ";AT 5,4;"        "; INK
0;"

950 FOR i=6 TO 14: FOR j=8 TO
23 STEP 5: PRINT INK 0;AT i,j
;" ": NEXT j: NEXT i
960 FOR i=8 TO 23: PRINT AT 15
,i; INK 0;"█": NEXT i
970 FOR I=6 TO 21: PRINT
PAPER 6;AT I,0;"        ";
AT I,24;"        ": NEXT I
980 FOR I=16 TO 21: PRINT
PAPER 6;AT I,8;"
        ": NEXT I
990 PRINT INK 1; PAPER 8;AT 2
,0;"PREMIO";AT 2,26;"PREMIO";
AT 3,1;"    ";AT 3,27;"
"
1000 PRINT INK 1; PAPER 8;AT 4
,2;"10";AT 4,27;"100";AT 6,2;"2
0";AT 8,2;"50";AT 7,27;"RAPA";A
T 8,25;"       "
1010 PRINT INK 1; PAPER 8;AT 2
0,6;"<SPACE> GIRA AS RODAS";AT
21,3;"HOLD SEGURA E NUDGE EMPUR
RA"
1020 PRINT AT 17,7;"      1
  2      3      ";AT 16,9;"
                    "
1030 PRINT AT 18,7;" 4=1&2 5=2&
3 6=1&3";AT 19,4; INK 7; PAPER
2;" HOLD : TECLAS DE 1 A 6 "
1040 RETURN
```

A linha 10 salta para a sub-rotina da linha 860, que desenha o caça-níqueis. As instruções ao jogador são colocadas na frente da máquina.

A linha 100 cria os blocos gráficos — UDG — na memória usando o comando **READ** para ler os valores contidos nas linhas **DATA** 20 a 90. Depois que o banco de blocos é criado, os caracteres das frutas aparecem nas linhas 140 a 150, de onde o programa retira dados para produzir as rodas giratórias. As variáveis **a$**, **b$** e **c$** representam as três rodas, e a ordem das frutas na rodas é determinada pela ordem dos blocos nas linhas **DATA**.

A TELA

```
200 GOSUB 1050
1050 PRINT AT 5,0;A$(4);"--- ";
AT 7,0;A$(4),A$(4);"--";AT 9,0;
A$(4);A$(4);A$(4),AT 10,0;A$(1)
;A$(1);A$(1)
1060 PRINT AT 11,0;C$(2);C$(2);
C$(2);AT 12,0;B$(1);B$(1);B$(1)
1070 PRINT AT 5,26,C$(1);C$(1);
C$(1);AT 6,26;C$(3);C$(3);C$(3)
;AT 9,26;C$(4);C$(4),C$(4)
1080 PRINT AT 7,10;A$(1);AT 10,
10,A$(2);AT 13,10,A$(3)
1090 PRINT AT 7,15;B$(1);AT 10,
15;B$(2);AT 13,15,B$(3)
1100 PRINT AT 7,20,C$(1);AT 10,
20;C$(2);AT 13,20;C$(3)
1110 PRINT AT 16,26; INK 2; PAP
ER 6;"NUDGE";AT 17,24,"TECLAS",
AT 15,25,"Q-W-E P";AT 18,25;"A-
S-D O"
1120 PRINT INK 0; PAPER 7,AT 1
2,25;"        ";AT 13,25,"
 ";AT 13,31;" ";AT 14,25;"
       ";AT 13,26; INK 2;"
   "
1130 PRINT PAPER 6, INK 0; INV
ERSE 1;AT 15,0;"TOTAL"; INVERSE
 0,AT 17,0;"$";AT 18,0;"C",AT 1
7,1; PAPER 7; BRIGHT 1;" 1";AT
18,1," 00"
1140 PRINT #1;AT 0,0;"      CACI
FE INICIAL = 1 DOLLAR"
1150 PAUSE 0
1160 PRINT #1;AT 0,0;"

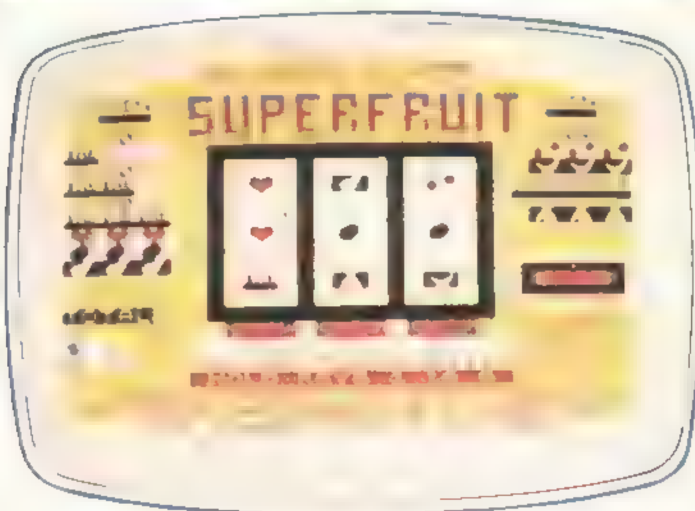
      "
1170 RETURN
```

A sub-rotina da linha 1050 desenha as rodas em sua posição inicial junto com alguns detalhes complementares. O jogador é informado que seu cacife inicial é de um dólar e que cada jogada custa dez centavos.

T

OS BLOCOS GRÁFICOS

```
10 PMODE 3,1:PCLS:CLS
20 DIM B(12),C(12),A(12),BR(12)
,S(12),PL(12),P(12),R1(15),R2(1
5),R3(15),W(9),H(29)
30 DRAW"BM16,0C2L2GLG4DG4D2R7FR
FR3ERER7U2H4UH4LH":PAINT(14,10)
:DRAW"BM17,2C1F4DF"
40 GET(0,0)-(31,15),B,G
50 DRAW"BM62,0C2L6GL6G2C4L3GLGL
GFDGLGLGLGFRFRFR3ERERER5FRFR3ER
EREHLHLHL5HEHLHLH":PAINT(48,8)
DRAW"BM41,8C1FRFRFRNFUR2UR2URBM
-4,-2HBM-5,7HBR17H"
60 GET(32,0)-(63,15),C,G
70 DRAW"BM80,0C3G8R17NH8BD2LNL1
5GLGLGLGL3NH3RFR7E":PAINT(80,4)
:PAINT(80,13)
80 GET(64,0)-(95,15),A,G
90 DRAW"BM96,0C4R30BD2L30DR30BD
6UBU2HL4D2NR4D2BL7UJHL3GDNR4D2B
L12R4EHEHL5D2NR2D2BD2R30DL30BD2
R30"
100 GET(96,0)-(127,15),BR,G
110 DRAW"BM148,0C2GL3GC4LHLGL3G
4RF3RF5RFR3ERE4UER2E3LH3LHLGL5G
L":PAINT(144,8):DRAW"BM138,3C1R
```





O caça-niqueis "Superfruit" é bem atraente no Acorn, micro que não existe no Brasil. À direita, como ele aparece no Spectrum.

```
BR13RBDBL10LBDBL5LBDBR8RBR9RBDB
L4LBL13LBDBR10RBR4BDLBL9LBDBR4R
BG2LBR6R"
120 GET(128,0)-(159,15),S,G
130 DRAW"BM186,0C3L2GL5DGR6DF4D
G2LG3LGL5H6E5R2":PAINT(176,8):D
RAW"BR6BDC1D2F"
140 GET(160,0)-(191,15),PL,G
150 DRAW"BM218,0C2L4DL3D3F4DF4L
GLGL13HLHUE7REU2":PAINT(208,8)
DRAW"BM211,6C1DF2"
160 GET(192,0)-(223,15),P,G
```

O nosso caça-niqueis e desenhado em **PMODE3**. Os comandos **GET** e **PUT** são utilizados para desenhar os símbolos das frutas nas rodas giratórias.

As matrizes que contêm os símbolos são dimensionadas na linha 20 **B** para o sino, **C** para a cereja, **A** para a bolota de carvalho, **BR** para a barra, **S** para o morango, **PL** para a ameixa e **P** para a pêra. As matrizes **R1**, **R2** e **R3** representam o conteudo das três rodas; **W** contém os valores dos prêmios e **H** é usada para segurar as rodas, impedindo que girem livremente.

AS INSTRUÇÕES

```
170 BS=CHR$(128)·CLS:PRINT @9,B
$"superfruit"BS
180 PRINT"VOCE TEM 1 DOLAR.CADA
 JOGADA     CUSTA .10,ATE ACABAR
 O DINHEIRO"
190 PRINT"controles:":PRINT "<S
PACE>"TAB(3)"-GIRA RODAS/APOSTA
":PRINT"<1>"TAB(3)"-LIBERA RODA
S"
200 PRINT"<2>"TAB(3)" SEGURA RO
DA ESQUERDA":PRINT"<3>"TAB(3)"-
SEGURA RODA DO MEIO":PRINT"<4>"
TAB(3)"-SEGURA RODA DIREITA"
210 PRINT "<5>"TAB(3)"-EMPURRA
RODA ESQ  P/ CIMA"·PRINT"<6>"TA
B(3)"-EMPURRA RODA CENTRAL P/CI
MA":PRINT"<7>"TAB(3)"-EMPURRA R
ODA DIR. P/ CIMA"
220 PRINT "<8>"TAB(3)"-EMPURRA
RODA ESQ.P/BAIXO":PRINT"<9>"TAB
(3)"-EMPURRA RODA CENT  P/BAIXO
":PRINT "<0>"TAB(3)"-EMPURRA RO
DA DIR.P/BAIXO":PRINT"<ENTER>"T
AB(3)"-RECOLHE O PREMIO";
```

As linhas 170 a 220 escrevem as instruções na tela de textos.

A TELA INICIAL

```
230 IF INKEY$<>" " THEN 230
240 FOR A=0 TO 15:READ R1(A),R2
(A),R3(A):NEXT
250 DATA 0,1,2,3,5,6,6,2,0,5,3,
4,4,6,5,6,4,3,1,2,0,3,0,5,2,1,4
,6,5,1,0,6,6,1,4,3,2,0,1,5,3,2,
4,6,6,6,4,5
260 FOR A=0 TO 9·READW(A):NEXT
270 DATA 200,150,100,80,60,40,3
0,20,10,0
280 GOSUB 4000
290 SCREEN 1,0:PCLS 3:DRAW"BM84
,4C2S20LDRDLBR2NU2RU2BRND2RDLBE
BRNRDNRDRBRU2RDLFBRUNRURBRND2RD
LFBRNU2RU2BRD2BR2U2LR2"
300 FOR K=0 TO 2:LINE (40+64*K,
20)-(87+64*K,115),PRESET,BF:NEX
T
310 FOR K=0 TO 2:DRAW"BM"+STR$(
40+64*K)+",124S16R12D4L12U4BFD2
BRUNLUBR2RD2LU2BR3D2RBR2U2S8RFD
2GL"·NEXT
320 GET(38,122)-(91,143),H,G
330 COLOR 4:FOR K=1 TO 5:LINE(1
0+K*16,158)-(21+K*16,169),PSET,
BF·NEXT
340 DRAW"BR30C1S24U2F2U2BRD2RU2
BRD2S8RE2U2H2LS24BR3LD2RUBENRDN
RDR"
350 GOTO 350
4000 CLS:PRINT @11,"premios"
4010 PRINT @65,"BARRA  BARRA  B
ARRA":PRINT" BOLOTA BOLOTA BOLO
TA":PRINT" AMEIXA AMEIXA AMEIXA
"
4020 PRINT "MORANGO MORANGO MOR
ANGO":PRINT" PERA   PERA   PERA
":PRINT" SINO   SINO   SINO"
4030 PRINT" CEREJA CEREJA CEREJ
A":PRINT" SINO   SINO      -":PR
INT" CEREJA CEREJA   -":PRINT"
CEREJA   -      -"
4040 FOR A=0 TO 9:IF A<7 THEN P
RINT @89+A*32,USING"$$#.##";W(A
)/100;:GOTO 4060
4050 PRINT @89+A*32,USING"$$#.#
#";W(A-1)/100
4060 NEXT
4070 PRINT @416,"PRESSIONE <SPA
CE> PARA CONTINUAR"
4080 IF INKEY$<>" " THEN 4080
4090 RETURN
```

As linhas 240 e 250 colocam as rodas na tela — cada número representa uma das frutas. As linhas 260 e 270 acertam os valores dos prêmios. A linha 280 salta para a sub-rotina que começa na linha 4000; esta exibe os resultados premiados e seus valores.

As linhas 290 e 340 desenham a tela inicial. É a linha 290 que liga a tela de alta resolução, permitindo o surgimento do caça-níqueis na tela.

O programa listado a seguir cuida especificamente da parte gráfica básica do jogo caça-níqueis.

```
5 CLEAR 5000
10 SCREEN 1:COLOR 1,11,15·KEY O
FF
20 DATA 14,31,31,31,31,15,3,1,5
6,252,252,252,252,248,224,192
30 DATA 49,42,51,42,50,255,255,
255,152,84,216,84,84,255,255,25
5
40 DATA 3,4,8,28,62,62,62,28,28
,190,125,62,28,0,0,0
50 DATA 0,0,16,28,15,7,3,0,4,12
,26,56,248,240,224,0
60 DATA 0,0,7,15,31,31,15,7,8,2
```

```
48,240,240,224,224,192,128
70 DATA 1,3,3,7,15,15,24,1,128,
192,192,224,240,240,24,128
80 DATA 3,7,15,31,31,31,31,15,2
24,240,240,240,240,224,192,128
90 DATA 8,8,8,8,8,73,42,28,8,16,5
6,84,146,16,16,16,16
100 FOR I=15*8 TO 30*8 STEP 8
110 FOR J=0 TO 7
120 READ A:VPOKE BASE(7)+I*8+J,
A
130 NEXT J,I
140 FOR I=15 TO 31
150 READ A:VPOKE BASE(6)+I,A
160 NEXT
170 DATA 143,143,31,31,111,111,
191,191,63,63,223,223,159,159,3
1,31,0
180 DIM A(24,2),B(24,2),C(24,2)
190 FOR I=0 TO 11:READ A(I,1),A
(I,2):A(I+12,1)=A(I,1):A(I+12,2
)=A(I,2):NEXT I
200 FOR I=0 TO 11:READ B(I,1),B
(I,2):B(I+12,1)=B(I,1):B(I+12,2
)=B(I,2):NEXT I
210 FOR I=0 TO 11:READ C(I,1),C
(I,2):C(I+12,1)=C(I,1):C(I+12,2
)=C(I,2):NEXT I
220 DATA 200,208,120,128,136,14
4,152,160,168,176,184,192,200,2
08,216,224,168,176,136,144,216,
224,120,128
230 DATA 216,224,136,144,200,20
8,152,160,216,224,168,176,184,1
92,120,128,200,208,152,160,168,
176,120,128
240 DATA 120,128,200,208,184,19
2,152,160,216,224,120,128,136,1
44,168,176,200,208,184,192,152,
160,216,224
860 PRINT ""
870 FOR I=0 TO 31: VPOKE BASE(5
)+I,254:NEXT
880 LOCATE 4,0:PRINT "10 CENTAV
OS A JOGADA"
890 LOCATE 2,2:PRINT " ";CHR$(
254);" [illegible]
[illegible] "
900 LOCATE 2,3:PRINT " ";CHR$(
254);" [illegible] ";CHR$(254);"[illegible]
[illegible] ";CHR$(254);"[illegible]";CHR$
(254);
910 LOCATE 2,4:PRINT " ";CHR$(
254);" [illegible] ";CHR$(254);"[illegible]
[illegible] ";CHR$(254);"[illegible] ";CHR
$(254);" "
920 LOCATE 2,5:PRINT " [illegible]
[illegible] "
970 FOR I=6 TO 14:LOCATE 8,I:PR
INT STRING$(12,254):NEXT
990 LOCATE 1,4:PRINT "10":LOCA
TE 0,6:PRINT "20":LOCATE 1,8:P
RINT "50":LOCATE 25,4:PRINT "1
00":LOCATE 25,7:PRINT "RAPA":
LOCATE 24,8:PRINT "[illegible]":V
POKE BASE(5)+287,197
1000 LOCATE 8,15:PRINT " [illegible]
[illegible]";:LOCATE 10,16:PRINT
"1     2     3";
1010 LOCATE 6,17:PRINT "4=1&2 5
=2&3 6 =&3"  LOCATE 6,18:PRINT
"HOLD TECLAS 1 A 6";
1020 LOCATE 1,19:PRINT "BARRA D
E ESPAÇO PARA RODAR"
1030 LOCATE 7,20:PRINT "HOLD SE
GURA  E NUDGE EMPURRA";
1050 FOR I=1 TO 7:VPOKE BASE(5)
+159+I,A(2,I):NEXT:LOCATE 0,5:P
RINT STRING$(4,254);
1060 FOR J=0 TO 1:FOR I=1 TO 2:
VPOKE BASE(5)+223+I+J*2,A(2,I):
NEXT I,J:LOCATE 2,7:PRINT CHR$(
254);CHR$(254);
1070 FOR J=0 TO 2:FOR I=1 TO 2
VPOKE BASE(5)+287+I+J*2,A(1,I)
NEXT I,
1072 FOR J=0 TO 2:FOR I=1 TO 2
VPOKE BASE(5)+185+I+J*2,A(3,I)
NEXT I,J
1074 FOR J=0 TO 2:FOR I=1 TO 2:
VPOKE BASE(5)+27[illegible]+J*2,A(8,I)
NEXT I,J
1076 FOR J=0 TO 2:FOR I=1 TO 2
VPOKE BASE(5)+313+I+J*2,A(6,I
NEXT I,J
1080 FOR J=0 TO 2:FOR I=1 TO 2:
VPOKE BASE(5)+319+I+J*2,A(5,I):
NEXT I,J
1090 FOR J=0 TO 2:FOR I=1 TO 2:
VPOKE BASE(5)+351+I+J*2,A(7,I):
NEXT I,J
1100 FOR I=235 TO 500 STEP 96:F
OR J=1 TO 2:VPOKE BASE(5)+I+J-1
,A((I-235)/96+1,J):NEXT J,I
1110 FOR I=239 TO 500 STEP 96:F
OR J=1 TO 2:VPOKE BASE(5)+I+J-1
,B((I-239)/96+1,J):NEXT J,I
1120 FOR I=243 TO 500 STEP 96:F
OR J=1 TO 2:VPOKE BASE(5)+I+J-1
,C((I-243)/96+1,J):NEXT J,I
1130 LOCATE 22,13:PRINT "TECLAS
";:LOCATE 22,14:PRINT "NUDGE";:
LOCATE 21,12:PRINT "Q-W-E  ":LO
CATE 21,15:PRINT "A-S-D  "
1140 LOCATE 0,13:PRINT "TOTAL";
 LOCATE 0,14:PRINT "S";:LOCATE
0,15:PRINT "C";:LOCATE 2,14:PRI
NT "1";:LOCATE 2,15:PRINT "00";
1150 LOCATE 1,22:PRINT "CACIFE
INICIAL    1 DOLLAR";
```

Os blocos gráficos que se encontram na listagem podem ser obtidos via teclas **SHIFT, CODE e GRAPH.**

O programa é executado na tela de textos de 32 colunas. Ao mesmo tempo, os símbolos que representam as frutas são desenhados como se fossem caracteres. Os padrões correspondentes devem ser transferidos das linhas **DATA** 20 a 90 para a tabela de padrões na VRAM — linha 120. Os caracteres são coloridos pela linha 150, que coloca os códigos apropriados na tabela de cores da VRAM — **BASE(6)**.

Três matrizes são utilizadas para representar as rodas giratórias: **A, B** e **C**. Elas são dimensionadas na linha 180. A ordem dos símbolos nas rodas é obtida pelas linhas 190 a 210 nas linhas **DATA** 220 a 240.

A máquina de frutas, com as instruções ao jogador, é desenhada pelas linhas 860 a 1060. Os resultados premiados aparecem na tela com o auxílio das linhas 1070 a 1090.

As rodas são desenhadas em sua posição inicial pelas linhas 1110 a 1120. E as linhas restantes dão mais algumas instruções.

No próximo artigo, veremos como colocar as rodas em movimento.



O programa listado a seguir cria os blocos gráficos e as telas com o caça-níqueis e as instruções.

```
10  HOME :E = 35000: HIMEM: E
20 F =  INT (E / 256): POKE 232
,E - F * 256: POKE 233,F
30  FOR I = E TO E + 41 + 16 *
32
40  READ A: POKE I,A
50  NEXT
60  SCALE= 1: ROT= 0
70  DATA  20 ,0 ,42 ,0 ,74 ,0 ,
106 ,0 ,138 ,0 ,170 ,0 ,202 ,0
,234 ,0 ,10 ,1 ,42 ,1 ,74 ,1 ,1
06 ,1 ,138 ,1 ,170 ,1 ,202 ,1 ,
234 ,1 ,10 ,2 ,42 ,2 ,74 ,2 ,10
6 ,2 ,138 ,2
80  DATA   0 ,72 ,105 ,77 ,218
,219 ,27 ,87 ,13 ,13 ,77 ,218 ,
251 ,31 ,87 ,13 ,13 ,77 ,218 ,2
51 ,31 ,87 ,9 ,13 ,77 ,218 ,251
 ,219 ,2 ,0 ,0 ,0
90  DATA   0 ,104 ,13 ,77 ,209
,251 ,219 ,51 ,13 ,13 ,13 ,141
,27 ,31 ,31 ,31 ,110 ,13 ,13 ,1
41 ,27 ,31 ,31 ,31 ,110 ,13 ,77
 ,209 ,219 ,27 ,31 ,6
100  DATA   0 ,40 ,109 ,73 ,213
 ,251 ,251 ,51 ,45 ,77 ,105 ,26
 ,255 ,251 ,51 ,45 ,77 ,105 ,21
8 ,219 ,219 ,42 ,45 ,45 ,45 ,21
3 ,63 ,63 ,63 ,55 ,0 ,0
110  DATA   0 ,104 ,41 ,109 ,20
9 ,251 ,251 ,87 ,13 ,45 ,77 ,21
8 ,223 ,31 ,119 ,13 ,77 ,169 ,2
19 ,219 ,155 ,45 ,45 ,45 ,173 ,
59 ,63 ,63 ,63 ,6 ,0 ,0
120  DATA   0 ,72 ,73 ,73 ,218
,223 ,219 ,74 ,73 ,73 ,218 ,223
 ,219 ,74 ,9 ,13 ,141 ,27 ,223
,251 ,106 ,13 ,77 ,209 ,219 ,27
 ,159 ,0 ,0 ,0 ,0 ,0
130  DATA   0 ,40 ,77 ,73 ,209
,219 ,219 ,115 ,73 ,73 ,209 ,21
9 ,219 ,83 ,77 ,73 ,209 ,219 ,2
51 ,83 ,13 ,13 ,77 ,218 ,219 ,2
23 ,2 ,0 ,0 ,0 ,0 ,0
140  DATA   0 ,72 ,73 ,73 ,218
,251 ,219 ,74 ,73 ,77 ,218 ,251
 ,223 ,74 ,105 ,77 ,218 ,251 ,3
1 ,87 ,77 ,105 ,209 ,219 ,251 ,
19 ,0 ,0 ,0 ,0 ,0 ,0
150  DATA   0 ,104 ,73 ,73 ,218
 ,219 ,251 ,110 ,77 ,73 ,218 ,2
7 ,31 ,31 ,110 ,13 ,77 ,209 ,25
1 ,31 ,31 ,110 ,77 ,105 ,218 ,2
```


SLOTS A FUN
CIRCUS CIRCUS
HOTEL·CASINO
FREE CIRCUS ACTS
ROOMS AVAILABLE
ALL NEW MIDWAY
LIVE TABLE GAMES
25¢ CRAPS 50¢ BLACKJACK
PENNY SLOTS
39¢ HOTDOG
44¢ HAMBURGER
GIFTS PACKAGE LIQUORS

```
7 ,223 ,51 ,0 ,0 ,0 ,0
160  DATA   0 ,72 ,73 ,73 ,218
,219 ,219 ,74 ,9 ,13 ,141 ,27 ,
31 ,31 ,159 ,13 ,13 ,13 ,141 ,2
7 ,31 ,31 ,31 ,110 ,13 ,13 ,141
,27 ,31 ,31 ,159 ,0
170  DATA   0 ,72 ,105 ,77 ,218
,251 ,31 ,87 ,13 ,13 ,77 ,218
,219 ,31 ,87 ,13 ,77 ,137 ,219
,27 ,31 ,87 ,77 ,73 ,209 ,219 ,
219 ,19 ,0 ,0 ,0 ,0
180  DATA   0 ,72 ,73 ,73 ,218
,251 ,219 ,74 ,105 ,77 ,218 ,25
1 ,31 ,87 ,73 ,73 ,209 ,27 ,31
,31 ,87 ,9 ,13 ,77 ,218 ,219 ,2
19 ,2 ,0 ,0 ,0 ,0
190  DATA   0 ,104 ,73 ,73 ,218
,219 ,251 ,110 ,13 ,77 ,209 ,2
51 ,31 ,31 ,78 ,73 ,73 ,218 ,31
,31 ,31 ,110 ,13 ,77 ,209 ,219
,219 ,51 ,0 ,0 ,0 ,0
200  DATA   0 ,72 ,73 ,73 ,218
,223 ,219 ,74 ,13 ,13 ,141 ,27
,31 ,31 ,31 ,110 ,13 ,13 ,141 ,
27 ,31 ,31 ,31 ,78 ,13 ,13 ,141
,27 ,31 ,223 ,19 ,0
210  DATA   0 ,72 ,105 ,77 ,218
,219 ,219 ,110 ,105 ,73 ,218 ,
251 ,31 ,87 ,13 ,13 ,77 ,218 ,2
51 ,31 ,87 ,13 ,77 ,137 ,219 ,2
19 ,187 ,0 ,0 ,0 ,0 ,0
220  DATA   0 ,40 ,45 ,45 ,45 ,
213 ,63 ,63 ,63 ,55 ,45 ,45 ,45
,173 ,59 ,63 ,63 ,63 ,78 ,73 ,
73 ,218 ,219 ,219 ,74 ,73 ,73 ,
218 ,219 ,219 ,2 ,0, 0
230  DATA    0 ,72 ,73 ,73 ,21
8 ,219 ,219 ,106 ,73 ,73 ,218 ,
219 ,251 ,110 ,77 ,73 ,218 ,31
,31 ,159 ,9 ,13 ,13 ,141 ,27 ,3
1 ,223 ,19 ,0 ,0 ,0
260  DIM R1(15),R2(15),R3(15)
270  FOR I = 0 TO 9: READ W(I):
 NEXT
280  DATA   2,1.5,1,.8,.6,.4,.3
,.2,.1,0
300  GOSUB 310: END
305  END
310  HOME : HGR2
320  DATA   54,20,40,20,40,40,5
4,40,54,60,40,60,0,0
330  DATA   60,20,60,60,74,60,7
4,20,0,0
340  DATA   80,60,80,20,94,20,9
4,40,80,40,0,0
350  DATA   114,20,100,20,100,4
0,110,40,100,40,100,60,114,60,0
,0
360  DATA   120,60,120,20,134,2
0,134,40,120,40,130,40,134,60,0
,0
370  DATA   154,20,140,20,140,4
0,150 ,40,140,40,140,60,0,0
380  DATA  160,60,160,20,174,20
,174,40,160,40,170,40,174,60,0,
0
390  DATA  180,20,180,60,194,60
,194,20,0,0
400  DATA   210,20,210,60,0,0
410  DATA   220,20,234,20,227,2
0,227,60,200,200
490  HCOLOR= 3
500  READ X,Y
510  HPLOT X,Y
520  READ X,Y: IF X = 0 AND Y =
 0 THEN 500
530  IF X = 200 AND Y = 200 THE
N 590
540  HPLOT  TO X,Y
550  GOTO 520
590  HCOLOR= 3
600  FOR I = 70 TO 150
610  HPLOT 60,I TO 210,I
620  NEXT I
630  HCOLOR= 0: FOR I = 80 TO 1
40
640  HPLOT 75,I TO 105,I
650  HPLOT 120,I TO 150,I
660  HPLOT 165,I TO 195,I
670  NEXT
680  RETURN
700  HOME : PRINT  TAB( 15);"SU
PERFRUIT": PRINT : PRINT
710  PRINT "VOCE COMECA COM $1.
 CADA JOGADA CUSTA   DEZ CENTAV
OS. O JOGO DURA ATE SEU        D
INHEIRO ACABAR."
720  PRINT : INVERSE : PRINT "C
ONTROLES:": NORMAL : PRINT
730  PRINT "<ESPACO>"; TAB( 10)
;"GIRA AS RODAS/APOSTA": PRINT
"<1>"; TAB( 10);"LIBERA AS RODA
S"
740  PRINT "<2>" TAB( 10)"SEGUR
A RODA ESQUERDA": PRINT "<3>" T
AB( 10)"SEGURA RODA DO MEIO": P
RINT "<4>" TAB( 10)"SEGURA RODA
 DIREITA"
750  PRINT "<5>" TAB( 10)"EMPUR
RA RODA ESQ.PARA CIMA": PRINT "
<6>" TAB( 10)"EMPURRA RODA CENT
RAL PARA CIMA": PRINT "<7>" TAB
( 10)"EMPURRA RODA DIR.PARA CIM
A"
760  PRINT "<8>" TAB( 10)"EMPUR
RA RODA ESQ.PARA BAIXO": PRINT
"<9>" TAB( 10)"EMPURRA RODA CEN
TRAL P/ BAIXO": PRINT "<0>" TAB
( 10)"EMPURRA RODA DIR PARA BAI
XO": PRINT "<ENTER>" TAB( 10)"R
ECOLHE PREMIO";
765  PRINT : PRINT "<ESPACO> PA
RA CONTINUAR ".
770  GET A$: IF A$ <  > " " THE
N 700
775  RETURN
780  FOR A = 0 TO 15: READ R1(A
),R2(A),R3(A): NEXT
790  DATA   0,1,2,3,5,6,6,2,0,5
,3,4,4,6,5,6,4,3,1,2,0,3,0,5,2,
1,4,6,5,1,0,6,6,1,4,3,2,0,1,5,3
,2,4,6,6,6,4,5
820  RETURN
4000  HOME : PRINT  TAB( 10);"V
ALORES PREMIADOS": PRINT : PRIN
T
4010  PRINT "BARRA  BARRA  BARR
A"; TAB( 30);W(0)
4020  PRINT : PRINT "BOLOTA BOL
OTA BOLOTA"; TAB( 30);W(1)
4030  PRINT : PRINT "AMEIXA AME
IXA AMEIXA"; TAB( 30);W(2)
4040  PRINT : PRINT "MORANGO MO
RANGO MORANGO"; TAB( 31);W(3)
```

MICRO DICAS

CUIDADOS ESPECIAIS

Verifique se as três partes do comando **DATA** ligadas aos objetos são lidas na matriz correta. Se tentarmos colocar uma cadeia de caracteres de um comando **DATA** em uma matriz numérica, receberemos uma mensagem de erro ou uma descrição curta onde esperávamos por uma longa.

Tenha muito cuidado ao combinar a ordem das partes do comando **DATA** com a ordem das matrizes nos comandos **READ**, pois o mesmo problema pode ocorrer. A ordem é: posição, descrição curta e descrição longa.

Faça um ''teste de mesa'' na sua aventura, depois de ter colocado os objetos, para verificar se eles aparecem nas posições corretas.

Use seu mapa quando checar os objetos, assegurando-se, assim, de que não está perdendo nenhum.

```
4050  PRINT : PRINT "PERA    PER
A   PERA"; TAB( 31);W(4)
4060  PRINT : PRINT "SINO    SIN
O   SINO"; TAB( 31);W(5)
4070  PRINT : PRINT "CEREJA CER
EJA CEREJA"; TAB( 31);W(6)
4080  PRINT : PRINT "SINO    SIN
O", TAB( 31);W(7)
4090  PRINT : PRINT "CEREJA CER
EJA"; TAB( 31);W(7)
4100  PRINT : PRINT "CEREJA"; T
AB( 31);W(8)
4110  GET A$: RETURN
```

A primeira parte do programa, que vai da linha 10 à linha 230, cria uma tabela de figuras móveis no topo da memória do micro. Essa figuras são os símbolos das frutinhas, que podem então ser desenhadas por intermédio do comando **DRAW**.

As linhas 320 a 670 desenham a máquina caça-níqueis na tela de alta resolução (nenhuma fruta aparecerá por enquanto). As linhas **DATA** de 320 a 410 são usadas para escrever a expressão "SUPER-FRUIT".

Uma tela com instruções destinadas ao jogador é criada pelas linhas que vão de 700 a 770. Outra tela, com os resultados que valem pontos, é escrita pelas linhas 4000 a 4100.

Três matrizes são utilizadas para representar as rodas: R1, R2 e R3. A ordem dos símbolos nas rodas é determinada pela linha **DATA** 790.

No próximo artigo, as rodas serão colocadas em movimento.

PROGRAMAÇÃO GRÁFICA DE CURVAS

- IDENTIFICAÇÃO DAS CURVAS
- UMA ELIPSE NO COTIDIANO
- NADO PARABÓLICO
- CÍRCULOS E POLÍGONOS
- GRÁFICOS FEITOS DE CURVAS

No primeiro artigo sobre seções cônicas, mostramos como desenhar círculos, elipses, parábolas e hipérboles no micro. Agora, veremos como incorporá-los aos programas.

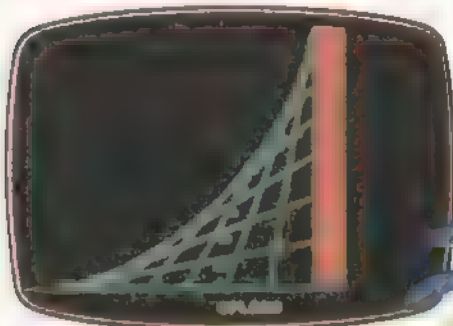
Como vimos em alguns exemplos do artigo anterior (página 801), as curvas cônicas estão no nosso dia-a-dia, por toda a parte, e muitas vezes nem chegamos a percebê-las. Mas não é difícil identificá-las: basta recorrer às equações. Se, ao calcularmos a posição de um objeto em movimento, verificarmos que sua coordenada **X** é dada por **A*COS T** e sua coordenada **Y**, por **A*SIN T** (onde **A** é uma distância fixa e **T**, um ângulo que varia), ficará absolutamente evidente que estamos lidando com a equação de um círculo.

Por outro lado, algumas vezes é mais fácil analisar o modo como algo se move do que calcular suas equações. Se, por exemplo, observarmos que a distância de um objeto a um ponto é sempre constante, saberemos que se trata de um círculo, sem precisarmos calcular sua equação. Esse tipo de análise também se aplica a outras curvas.

ELIPSE, PARÁBOLA, HIPÉRBOLE

Uma elipse não difere muito de um círculo e, assim como este, é de fácil identificação. Ela é desenhada quando

À medida que a escada escorrega, o balde preso a ela traça parte de uma elipse.



um ponto se move de tal maneira que a sua distância a um dos focos somada à sua distância ao outro foco permanece sempre constante.

Uma parábola, por sua vez, é desenhada quando um ponto se move de tal modo que a sua distância a um ponto fixo é igual à distância perpendicular a uma linha fixa. O ponto fixo corresponde ao foco da parábola. A linha fixa, chamada de *diretriz*, é uma reta perpendicular ao eixo de simetria da parábola e que não corta a mesma — ou seja, está fora dela. Como já vimos, a distância da diretriz a um ponto qualquer da curva é igual à distância desse mesmo ponto até o foco.

A descrição da hipérbole também é simples: ela é obtida quando um ponto se move e a distância dele a um ponto fixo menos sua distância a outro ponto fixo permanece constante. Os pontos fixos, um em cada metade da hipérbole, são seus focos.

Os programas apresentados a seguir demonstram os dois métodos de identificação de curvas — isto é, o de equações e o de observação do comportamento de um ponto em movimento. Uma vez percebida a presença de um certo tipo de curva num programa, fica bem mais fácil utilizá-lo.

UMA ESCADA ESCORREGANDO

O primeiro programa mostra parte de uma elipse obtida durante um evento comum: uma escada escorregando numa parede. Observamos que um balde preso à escada traça parte de uma elipse à medida que esta escorrega.

Suponhamos que a escada tem um comprimento de oitocentas unidades e que o balde está preso a quinhentas unidades do pé da escada. A posição do balde será **X = -300*COS (ângulo)** e **Y = 500*SIN (ângulo)**, o que sabemos corresponder à equação de uma elipse.



```
10  HOME :C =  ATN (1) / 45
15  HGR : HCOLOR= 3
20  GOSUB 250
30  GOSUB 50
40  GOTO 40
50  REM  ESCADA
60  FOR A = 80 TO 0 STEP  - 10
70  FOR T = 0 TO 250: NEXT
80  HPLOT 230 - 150 *  COS (C *
 A),150 TO 228,150 - 150 *  SIN
 (C * A)
110 X =  - 56 *  COS (C * A)
120 Y = 90 *  SIN (C * A)
130  GOSUB 190
140  PRINT  CHR$ (7)
150  NEXT
190  REM  BALDE
200  IF Y = 0 THEN  VTAB (22):
HTAB (22): PRINT "SPLASH"
210  HPLOT 228 + X,154 - Y TO 2
28 + X,150 - Y
220  HPLOT  TO 232 + X,150 - Y
230  HPLOT  TO 232 + X,154 - Y
240  HPLOT  TO 228 + X,154 - Y:
 RETURN
250  REM  PAREDE
260 OX = 230
270  HPLOT OX,0 TO OX,150
280  HPLOT OX + 8,0 TO OX + 8,1
50
290  HPLOT OX + 16,0 TO OX + 16
,150
300  FOR Y = 0 TO 150 STEP 10:
HPLOT OX,Y TO OX + 16,Y
310  NEXT
320  RETURN
```



```
10 LET wall=240: LET ladder=
60: LET bucket=190
20 GOSUB wall
30 GOSUB ladder
35 FLASH 0
40 GOTO 40
60 FOR a=80 TO 0 STEP -10
70 PAUSE 25: LET r=a/(180/PI)
80 PLOT ox-150*COS (r),oy
90 DRAW ox-(ox-150*COS (r)),
oy+150*SIN (r)
110 LET x=-60*COS (r)
120 LET y=90*SIN (r)
130 GOSUB bucket
140 SOUND .1,a/2-15
150 NEXT a
160 FLASH 1: PRINT AT 10,5;"SP
LASH"
170 RETURN
190 PLOT ox+x,oy+y+5: DRAW 0,-
2
200 FOR n=oy+y TO oy+y+2: PLOT
ox+x-2,n: DRAW 4,0
210 NEXT n
220 RETURN
240 BORDER 0: INK 7: PAPER 0:
CLS
250 LET ox=232: LET oy=8
260 FOR y=1 TO 20: PRINT
PAPER 2;AT y,29;"  "
270 NEXT y
280 FOR y=oy-1 TO 165 STEP 16:
PLOT ox,y
290 DRAW 16,0: PLOT ox,y+8:
DRAW 16,0: PLOT ox+8,y+8: DRAW
0,8
300 NEXT y
310 PLOT  INK 4;ox+8,oy-1:
DRAW  INK 4;-232,0
320 RETURN
```



```
10 COLOR 15,4,5:SCREEN2
15 C=ATN(1)/45
20 GOSUB 230:REM<parede>
30 GOSUB 50:REM<escada>
40 GOTO 40
50 FOR AN=80 TO 0 STEP-10
90 LINE(230-150*COS(C*AN),150)-
(228,150-150*SIN(C*AN)),15
110 X=-56*COS(C*AN)
120 Y=90*SIN(C*AN)
130 GOSUB 200:REM<balde>
140 FORT=0 TO 500:NEXT
150 NEXT
160 RETURN
200 IF Y=0 THEN Y=4:DRAW"BM160,
156C1S16LDRDLBR2U2RDLBEBRD2RBRU
2RDNLDBRRULURBRD2BRUNLUC4"
210 LINE(228+X,154-Y)-(232+X,15
0-Y),15,BF
220 RETURN
230 LINE(230,0)-(255,150),6,BF
250 FOR Y=0 TO 150 STEP 10
260 LINE(230,Y)-(255,Y),10
270 NEXT
```



```
280 FOR Y=0 TO 150 STEP 20
290 LINE(243,Y)-(243,Y+10),10
300 NEXT
310 LINE(0,151)-(255,191),3,BF
320 RETURN
```



```
10 PMODE 3,1:PCLS:SCREEN 1,0:C=
ATN(1)/45
20 GOSUB 230
30 GOSUB 50
40 GOTO 40
50 FOR AN=80 TO 0 STEP -10
70 COLOR 4,2
90 LINE(230-150*COS(C*AN),150)-
(228,150-150*SIN(C*AN)),PRESET
110 X=-56*COS(C*AN)
120 Y=90*SIN(C*AN)
130 GOSUB 200
140 FOR T=0 TO 500:NEXT
150 NEXT
160 RETURN
200 IF Y=0 THEN Y=4:DRAW"BM160,
156C2S16LDRDLBR2U2RDLBEBRD2RBRU
2RDNLDBRRULURBRD2BRUNLUC4"
210 LINE(228+X,154-Y)-(232+X,15
0-Y),PSET,BF
220 RETURN
230 LINE(230,0)-(255,150),PSET,
BF
240 COLOR 2
250 FOR Y=0 TO 150 STEP 10
260 LINE(230,Y)-(255,Y),PSET
270 NEXT
280 FOR Y=0 TO 150 STEP 20
290 LINE(243,Y)-(243,Y+10),PSET
300 NEXT
310 COLOR 3:LINE(0,151)-(255,19
1),PSET,BF
320 RETURN
```

O programa compõe-se de três rotinas principais — a que desenha a parede, a que desenha a escada e a que desenha o balde. A parede é traçada primeiro, entre as linhas 230 (250 no Apple) e 320. Depois, a rotina entre as linhas 50 e 160 desenha a escada em nove posições diferentes a intervalos de 10 graus. Essa rotina chama a seguinte, que fica entre as linhas 190 e 220 (240 no Apple) e tem a função de desenhar um balde para cada posição da escada. As coordenadas do balde são calculadas nas linhas 110 e 120, e, como vimos, elas formam a equação de uma elipse. As posições anteriores do balde e da escada não são apagadas, para que se perceba mais facilmente que o caminho percorrido pelo balde realmente faz parte de uma elipse.

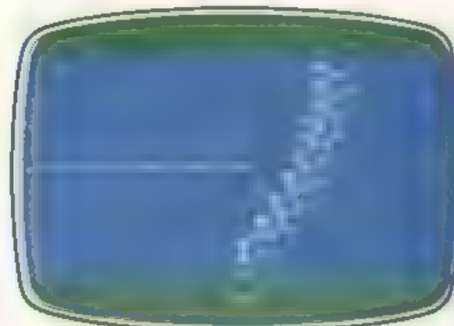
NADO PARABÓLICO

Imagine o que acontece quando um nadador tenta atravessar um rio de águas rápidas. Mesmo que ele tenha como alvo um determinado ponto na margem oposta, a correnteza certamente o desviará um pouco do seu destino. Caso a velocidade da correnteza se equipare à do nadador, a distância do desvio será igual à metade da largura do rio, e o efeito conjunto das duas velocidades fará com que o nadador percorra um caminho parabólico.

Para entendermos a razão por que isso ocorre, devemos pensar em termos de velocidade. O nadador visa sempre um ponto na outra margem e nada com uma velocidade **V**, enquanto o rio corre paralelamente à margem, com a mesma velocidade **V**. Combinando os dois valores, obtemos a velocidade real do nadador em relação à margem.

Esse raciocínio baseia-se na regra do paralelogramo de forças, empregada em problemas de física. Ela é usada na construção de uma parábola, onde a distância do nadador ao foco (ponto que ele visa na outra margem) é igual à dis-

Um nadador que se move com a velocidade da correnteza do rio percorre uma trajetória parabólica.



tância dele até a diretriz (distância percorrida pelo rio no mesmo intervalo de tempo).



```
10  HGR2 : HCOLOR= 3
20  GOSUB 50
30  GOSUB 200
40  GOTO 40
50  REM  DESENHA RIO
70  FOR LY = 0 TO 38: HPLOT 0,L
Y TO 279,LY: NEXT
80  FOR LY = 153 TO 191: HPLOT
0,LY TO 279,LY: NEXT
90  RETURN
200  FOR T =  - 1 TO  - .05 STE
P .1
210 X =  - 60 * T * T:Y = 120 *
 T
220 AN =  ATN ((X + 60) /  - Y)

230  GOSUB 300
240 FX =  INT (130 + Y): HPLOT
FX,95 TO FX + 8,95
245  HPLOT FX + 4,91 TO FX + 8,
95
250  HPLOT  TO FX + 4,99
255  NEXT
260  RETURN
300 XC = 187 + X:YC = 33 - Y
310 XX = 0:YY = 6: GOSUB 420
320  HPLOT  INT (XC + XT), INT
(YC + YT)
330  RESTORE
340  FOR N = 1 TO 16
350  READ XX,YY
360  GOSUB 420
370  HPLOT  TO XC + XT + .5,YC
- YT + .5
380  NEXT
390  RETURN
410  DATA  0,6,-6,0,0,6,6,0,0,6
,0,14,6,12,0,14,-6,12,0,14,0,18
,2,18,2,22,-2,22,-2,18,0,18
420 XT = XX *  COS (AN) - YY *
 SIN (AN)
430 YT = XX *  SIN (AN) + YY *
 COS (AN)
440  RETURN
```

MICRO DICAS

CONDIÇÕES DE VISUALIZAÇÃO

Várias providências podem ser tomadas para aperfeiçoar as condições de visualização do vídeo. Por exemplo, o esforço visual será diminuído com o uso de caracteres claros sobre fundo escuro. Assim, procure "sintonizar" pouco a pouco os controles de contraste e intensidade, até melhorar a definição dos caracteres.

Experimente posicionar sua mesa de trabalho de tal maneira que a tela não reflita a luz proveniente de janelas ou de luminárias. Incline ligeiramente a tela, dirigindo os reflexos para longe de seus olhos. Também é aconselhável cobrir a tela com um filtro anti-reflexivo (à venda em lojas de computadores).



```
 10 BORDER 0: PAPER 0: INK 7:
CLS
 20 REM rio
 60 LET parabola=190: LET swim
mer=300: LET rotate=430
 70 LET a$="
                                "
 80 FOR n=0 TO 3
 90 PRINT  PAPER 4;a$: NEXT n
100 FOR n=4 TO 18
110 PRINT  PAPER 1;a$: NEXT n
140 PRINT  PAPER 2; INK 6;AT 3
,15;"F";AT 19,15;"A";AT 3,22;"
0"
150 GOSUB parabola
160 STOP
190 LET ox=187: LET oy=150
200 FOR t=-1 TO -0.05 STEP 0.1
210 LET x=-60*(t*t): LET y=120
*t
220 LET a=ATN ((x+60)/-y)
230 PLOT ox-60+y,oy-60: DRAW
INK 7;10,0: DRAW  INK 7;-5,-5:
DRAW  INK 7;0,10: DRAW  INK 7;
5,-5
240 GOSUB swimmer
250 NEXT t
260 RETURN
300 LET ox=ox+x: LET oy=oy+y
310 LET x=0: LET y=6
320 GOSUB rotate
330 PLOT ox+xt,oy+yt
340 RESTORE 410
350 FOR n=1 TO 17
360 READ x,y
370 GOSUB rotate
380 DRAW xt,yt
390 NEXT n
400 LET ox=188: LET oy=150:
RETURN
410 DATA -3,0,0,3,3,0,0,-3,-2,
0,0,-3,-4,0,0,4,0,-4,8,0,0,4,0
,-4,-4,0,0,-4,-4,-4,4,4,4,-4
430 LET xt=x*COS (a)-y*SIN (a)
440 LET yt=x*SIN (a)+y*COS (a)
450 RETURN
```



```
10 COLOR 15,9,9:SCREEN2
20 GOSUB 50
30 GOSUB 200
40 GOTO 40
50 LINE(0,38)-(255,153),5,BF
60 DRAW"BM123,158C15S16ND2RDNLD
"
70 DRAW"BM123,25NRDNRD"
80 DRAW"BM180,25RD2LU2"
90 RETURN
200 FOR T=-1 TO -.05 STEP .1
210 X=-60*T*T:Y=120*T
220 AN=ATN((X+60)/-Y)
230 GOSUB 300
240 DRAW"BM"+STR$(INT(130+Y))+"
,95"+"C9R2NGH"
250 NEXT
260 RETURN
300 XC=187+X:YC=33-Y
310 XX=0:YY=6:GOSUB 420
320 DRAW"BM"+STR$(INT(XC+XT))+"
,"+STR$(INT(YC-YT))
330 RESTORE
340 FOR N=1 TO 16
350 READ XX,YY
360 GOSUB 420
370 LINE-(XC+XT+.5,YC-YT+.5),15
380 NEXT
390 RETURN
410 DATA 0,6,-6,0,0,6,6,0,0,6,0
,14,6,12,0,14,-6,12,0,14,0,18,2
,18,2,22,-2,22,-2,18,0,18
420 XT=XX*COS(AN)-YY*SIN(AN)
430 YT=XX*SIN(AN)+YY*COS(AN)
440 RETURN
```



```
10 PMODE 3,1:PCLS:SCREEN 1,0
20 GOSUB 50
30 GOSUB 200
40 GOTO 40
50 COLOR 3,2:LINE(0,38)-(255,15
3),PSET,BF
60 DRAW"BM123,158C4S16ND2RDNLD"
70 DRAW"BM123,25NRDNRD"
80 DRAW"BM180,25RD2LU2"
90 RETURN
200 FOR T=-1 TO .05 STEP .1
210 X=-60*T*T:Y=120*T
220 AN=ATN((X+60)/-Y)
230 GOSUB 300
240 DRAW"BM"+STR$(INT(130+Y))+"
,95"+"C2R2NGH"
250 NEXT
260 RETURN
300 XC=187+X:YC=33-Y
310 XX=0:YY=6:GOSUB 420
320 DRAW"BM"+STR$(INT(XC+XT))+"
,"+STR$(INT(YC-YT))
330 RESTORE
340 FOR N=1 TO 16
350 READ XX,YY
360 GOSUB 420
370 LINE -(XC+XT+.5,YC-YT+.5),P
RESET
380 NEXT
390 RETURN
410 DATA 0,6,-6,0,0,6,6,0,0,6,0
,14,6,12,0,14,-6,12,0,14,0,18,2
,18,2,22,-2,22,-2,18,0,18
420 XT=XX*COS(AN)-YY*SIN(AN)
430 YT=XX*SIN(AN)+YY*COS(AN)
440 RETURN
```

A primeira seção do programa, nas linhas 50 a 160, desenha o rio e as margens. A seção seguinte, nas linhas 170 a 290, usa a equação da parábola (linha 210) para calcular a posição do nadador. A rotina do nadador está entre as linhas 300 e 400. Ela recorre à rotina de rotação (linhas 420 a 450) para garantir que o nadador seja desenhado com o ângu-

lo correto, ou seja, para que esteja visando sempre o mesmo ponto na outra margem (que, neste caso, corresponde ao foco da parábola). O formato do nadador é definido pela linha **DATA** 410.

CÍRCULOS E POLÍGONOS

Para o computador, um círculo nada mais é do que um polígono de muitos lados. Quanto maior for o número de lados, mais lisa será a curva. Os usuários do Apple já devem ter percebido que temos usado polígonos para obter círculos, compensando a lamentável falta do comando **CIRCLE**. No programa a seguir, eles verão que as linhas 70, 75 e 80 simulam um comando **CIRCLE** para o desenho de um círculo de raio 70 e com centro em 127,95. Por uma questão visual, fizemos o raio na direção **Y** um pouco menor (65 em vez de 70). Caso contrário, o círculo ficaria oval, aparentando ser uma elipse.



```
10 PI = 4 *  ATN (1)
60  HOME : HGR  HCOLOR= 1
69  HPLOT 197,85
70  FOR TH = 0 TO 2 * PI STEP P
I / 16
75 CX = 70 *  COS (TH) + 127:CY
 = 65 *  SIN (TH) + 95
80  HPLOT  TO CX,CY: NEXT
90  VTAB (23): INPUT "FORNECER
ANGULO ";A
100 A = A *  ATN (1) / 45
110  GOSUB 260
130  HOME : VTAB (23): INPUT "J
OGA NOVAMENTE (S/N) ";ANS
140  IF ANS = "S" THEN 10
150  IF ANS < > "N" THEN 130
160  HOME : TEXT : END
260 TH = 2 * A
270 N = 0
280  HPLOT 185,95
300  HPLOT  TO 127 + 58 *  COS
(TH),95 - 58 *  SIN (TH)
310 TH = TH + 2 * A
330 N = N + 1
340  IF N < 15 THEN 300
350  RETURN
```



```
10 LET ox=100: LET oy=90
20 LET polygon=260
30 PAPER 0: INK 6
40 BORDER 0
50 CLS
80 PRINT  INK 7;AT 0,22;"Raio
 do"'TAB 22;"circulo e"'TAB
22;"82 unid."
90 PRINT  INK 4,AT 21,0;"Intr
oduza o angulo";
 95 CIRCLE 100,90,82
100 INPUT a
105 PRINT AT 21,0;"
"
110 GOSUB polygon
120 PRINT AT 21,6;"Outra vez ?
(S/N)"
130 LET a$=INKEY$: IF a$="s"
THEN  GOTO 20
140 IF a$="n" THEN  STOP
150 GOTO 130
260 LET a=a/(180/PI)
270 LET at=0
280 LET t=2*a
290 PLOT ox+82,oy
300 FOR n=0 TO 15
310 LET b=82*COS (t)+ox: LET c
=82*SIN (t)+oy
320 LET b=(b-(PEEK 23677)):
LET c=(c-(PEEK 23678)): DRAW b
,c
330 SOUND 0 01,(n*5)-20
340 LET t=t+2*a
350 NEXT n
360 RETURN
```



```
10 COLOR1,15,15
20 GOSUB 90
30 GOSUB 60
40 GOSUB 120
50 CLS:END
60 SCREEN2
70 CIRCLE(127,95),70,13:CIRCLE(
127,95),60,13:PAINT(127,30),13
80 GOTO 260
90 CLS:LOCATE9,10:INPUT"Forneça
 um ângulo ";A
100 A=A*ATN(1)/45
110 RETURN
120 SCREEN0:CLS
130 LOCATE5,10:INPUT"Joga novam
ente (s/n) ";ANS
140 IF ANS="S" OR ANS="s" THEN
RUN
150 IF ANS<>"N" AND ANS<>"n" TH
EN 120
160 RETURN
260 TH=2*A
270 N=0
280 DRAW"BM185,95"
300 LINE-(127+58*COS(TH),95-58*
SIN(TH)),1
310 TH=TH+2*A
320 PLAY"T255L64V"+STR$(15-N)+"
CAGF"
330 N=N+1
340 IF N<15 THEN 300
350 IF INKEY$="" THEN 350
360 RETURN
```



```
10 PMODE 3,1
60 PCLS:SCREEN 1,0
70 CIRCLE(127,95),70,4:CIRCLE(1
27,95),60,4:PAINT(127,30),4
80 FOR G=1 TO 2000:NEXT:COLOR 2
90 CLS:PRINT:INPUT"QUAL O ANGUL
O ";A
100 A=A*ATN(1)/45
110 SCREEN 1,0:GOSUB 260
120 IF INKEY$="" THEN 120
130 PRINT:PRINT:INPUT"NOVAMENTE
 (S/N) ";ANS
140 IF ANS="S" THEN 60
150 IF ANS<>"N" THEN 130 ELSE C
LS:END
260 TH=2*A
270 N=0
280 DRAW"BM185,95"
300 LINE-(127+58*COS(TH),95-58*
SIN(TH)),PSET
310 TH=TH+2*A
320 PLAY"T20V"+STR$(31-N*2)+"C"
330 N=N+1
340 IF N<15 THEN 300 ELSE RETUR
N
```



Efeitos muito interessantes podem ser obtidos pela sobreposição de curvas cônicas.

O programa desenha um círculo e pede que forneçamos o ângulo que a primeira linha formará com o lado do círculo. O ângulo **A** ou **a** é convertido para *Teta* (**TH** ou t) na linha 260 da rotina que desenha o polígono. O número de lados foi restrito a quinze (linha 340) para o diagrama não ficar parecendo apenas um emaranhado de linhas.

Se fornecermos um ângulo pequeno, o polígono se assemelhará a um círculo. Com ângulos maiores, a forma mais provável será a de uma estrela.

ARTE POR COMPUTADOR

O programa a seguir desenha hipérboles com diferentes excentricidades e, sobre elas, várias elipses. Podemos combinar outros tipos de curva para obter desenhos mais complexos — basta dar asas à imaginação.



```
10  HOME : HGR2 : HCOLOR= 3
20 C =  ATN (1) / 45:PI = 4 *
ATN (1)
30  GOSUB 70
40  GOSUB 255
```

```
50  GOTO 50
70  FOR E = 1 TO 1.25 STEP .02
100 A = 22:B = A *  SQR (E * E
 - 1)
110  HPLOT 128 +  INT (A /  COS
 ( - 80 * C)),95 -  INT (B *  T
AN ( - 80 * C))
130  FOR TH =  - 80 TO 80 STEP
20
140 X = A /  COS (TH * C)
150 Y = B *  TAN (TH * C)
160  HPLOT  TO 128 + X,95 - Y
170  NEXT
180  HPLOT 127 +  INT (A /  COS
 (100 * C)),95 -  INT (B *  TAN
 (100 * C))
190  FOR TH = 100 TO 260 STEP 2
0
200 X = A /  COS (TH * C)
210 Y = B *  TAN (TH * C)
220  HPLOT  TO 127 + X,95 - Y
230  NEXT TH,E
250  RETURN
255  HCOLOR= 2: HPLOT 222,95
260  FOR E = 1 TO .1 STEP  - .0
3
270  FOR AN = 0 TO 2 * PI STEP
PI / 16
280 CX = 95 *  COS (AN) + 127:C
Y = 90 * E *  SIN (AN) + 95
290  HPLOT  TO CX,CY: NEXT AN
300  NEXT E: RETURN
```



```
10 BORDER 0: PAPER 0: INK 7:
CLS
20 LET hiperbola=80
30 LET elipse=270
40 GOSUB hiperbola
50 GOSUB elipse
60 GOTO 60
80 LET ox=128: LET oy=87
90 FOR e=1 TO 2 STEP 0.05
100 LET a=22: LET b=a*(SQR (e^
2-1))
102 LET h=1
104 LET f=ox+(a/COS (-1.396))
106 LET g=oy+(b*TAN (-1.396))
108 IF g<0 THEN  LET h=0
110 PLOT  INVERSE 1; OVER 1;f,
h
120 IF g>0 THEN  PLOT  INK 6;f
,g
130 FOR t=-80 TO 80 STEP 20
135 LET r=t/(180/PI)
140 LET x=a/COS (r): LET y=b*
TAN (r)
142 LET c=oy+y: LET d=ox+x
150 IF h=0 THEN  LET d=f+g*(f-
d)/(c-g): PLOT d,h: LET c=0
160 IF c>175 THEN  LET d=d-((d
-PEEK 23677)*(c-175)/(c-PEEK
23678)): LET c=175
170 DRAW  INK 6,d-PEEK 23677,c
-PEEK 23678: NEXT t
172 LET f=ox+(a/COS (1.75))
174 LET g=oy+(b*TAN (1.75))
176 PLOT  INVERSE 1; OVER 1;f,
h
178 IF g<0 THEN  LET h=0
180 IF g>0 THEN  PLOT  INK 6;f
,g
190 FOR t=100 TO 260 STEP 20
195 LET r=t/(180/PI)
200 LET x=a/(COS (r)): LET y=b
*TAN (r)
202 LET c=oy+y: LET d=ox+x
204 IF h=0 THEN  LET d=f+g*(f-
d)/(c-g): PLOT d,h: LET c=0
206 LET h=1
210 IF c>175 THEN  LET d=0-((d
-PEEK 23677)*(c-175)/(c-PEEK
23678)): LET c=175
220 DRAW  INK 6;d-PEEK 23677,c
-PEEK 23678
230 NEXT t: NEXT e
250 RETURN
270 FOR e=0.5 TO 0.98 STEP
0.04
280 LET a=100: LET b=a*(SQR (1
-e^2))
290 PLOT ox+a,oy
300 FOR t=0 TO 360 STEP 10
305 LET r=t/(180/PI)
310 LET x=a*COS (r)
320 LET y=b*SIN (r)
330 DRAW x-(PEEK 23677)+ox,y-(
PEEK 23678)+oy
340 NEXT t: NEXT e
360 RETURN
```



```
10 COLOR 1,15,15:SCREEN 2
20 C=ATN(1)/45
30 GOSUB 70
40 GOSUB 260
50 GOTO 50
70 FOR E=1 TO 1.25 STEP .02
100 A=22:B=A*SQR(E*E-1)
110 DRAW"BM"+STR$(128+INT(A/COS
(-80*C)))+","+STR$(95-INT(B*TAN
(-80*C)))
130 FOR TH=-80 TO 80 STEP 20
140 X=A/COS(TH*C)
150 Y=B*TAN(TH*C)
160 LINE-(128+X,95-Y),1
170 NEXT
180 DRAW"BM"+STR$(127+INT(A/COS
(100*C)))+","+STR$(95-INT(B*TAN
(100*C)))
190 FOR TH=100 TO 260 STEP 20
200 X=A/COS(TH*C)
210 Y=B*TAN(TH*C)
220 LINE-(127+X,95-Y),1
230 NEXT TH,E
250 RETURN
260 FOR E=1 TO .1 STEP -.03
270 CIRCLE(127,95),95,8,,,E
280 NEXT:RETURN
```



```
10 PMODE 3,1:PCLS2:SCREEN 1,0
20 C=ATN(1)/45
30 GOSUB 70
40 GOSUB 260
50 GOTO 50
70 FOR E=1 TO 1.25 STEP .02
100 A=22:B=A*SQR(E*E-1)
110 DRAW"BM"+STR$(128+INT(A/COS
(-80*C)))+","+STR$(95-INT(B*TAN
(-80*C)))
130 FOR TH=-80 TO 80 STEP 20
140 X=A/COS(TH*C)
150 Y=B*TAN(TH*C)
160 LINE-(128+X,95-Y),PSET
170 NEXT
180 DRAW"BM"+STR$(127+INT(A/COS
(100*C)))+","+STR$(95-INT(B*TAN
(100*C)))
190 FOR TH=100 TO 260 STEP 20
200 X=A/COS(TH*C)
210 Y=B*TAN(TH*C)
220 LINE-(127+X,95-Y),PSET
230 NEXT TH,E
250 RETURN
260 FOR E=1 TO .1 STEP -.03
270 CIRCLE(127,95),95,3,E
280 NEXT:RETURN
```



A equação para um círculo também é usada para os polígonos.

Tanto a elipse quanto a hipérbole podem ser desenhadas com excentricidades diferentes — este é o papel que cabe à variável **E** ou **e**.

Para as elipses, **E** varia de 0 até 1, fazendo-as ir desde um círculo até uma reta. No programa, **E** varia de .5 a .98, o que garante que as elipses sejam suficientemente abertas.

Nas hipérboles, **E** pode variar de 1 até o infinito, mas, novamente, o programa restringe o intervalo, fazendo com que **E** varie de 1 a 2. Quanto maior for **E**, mais próxima a hipérbole estará de uma reta.

É muito fácil calcular a excentricidade de uma elipse ou de uma hipérbole. Para uma elipse de equações **X = A*COS T** e **Y = B*SIN T**, a fórmula será $E^2 = B^2/A^2 - 1$, como pode ser observado na linha 280 do programa para o Spectrum. Os outros micros não precisam calcular a variável **E**, que é simplesmente incorporada ao comando **CIRCLE** (MSX e TRS-Color) ou à simulação do mesmo (Apple, linha 280).

Para uma hipérbole cujas equações são do tipo **X = A/COS T** e **Y = B*TAN T**, temos $E^2 = 1 - B^2/A^2$. Uma versão adaptada dessa fórmula é utilizada na linha 100 do programa.

OS SEGREDOS DO SPECTRUM (1)

- COMO ESCREVER NAS LINHAS RESERVADAS DA TELA
- DELIMITAÇÃO DAS ÁREAS INDEPENDENTES DA TELA
- UMA TELA LISTRADA

Existem muitos recursos "ocultos" na ROM do ZX Spectrum. Conhecendo-os, você poderá empregar fantásticos truques de programação em BASIC, obtendo o máximo de seu computador.

O micro Sinclair ZX Spectrum ilustra muito bem o famoso ditado "tamanho não é documento". Os felizes usuários dessa popular maquininha, que já tiveram a oportunidade de conhecer mais a fundo seus recursos técnicos, podem testemunhar o quanto eles são poderosos — mais até do que os encontrados em muitos microcomputadores profissionais, maiores e mais caros.

Na série de artigos que iniciamos aqui, você verá de que modo pode explorar alguns "recursos secretos" do Spectrum e de seus compatíveis nacionais (como o TK-90X e o TK-95 da Microdigital) e internacionais.

Por que "secretos"? Como você mesmo terá oportunidade de verificar, tais recursos não estão documentados no Manual de Operação e, portanto, são desconhecidos da grande maioria dos usuários.

A TELA PROTEGIDA

É provável que você ainda não tenha descoberto a possibilidade de escrever nas duas linhas da parte de baixo da tela. Normalmente, não se usa o comando **PRINT** para escrever nessas linhas, pois elas são reservadas para a entrada de dados (**INPUT**), para a entrada e edição de programas e para a impressão de mensagens de erro do interpretador BASIC. É muito fácil, porém, modificar um comando **PRINT** de tal maneira que se possa utilizar essa área independente da tela:

```
10 PRINT 0."Mensagem"
```

O comando funciona evidentemente dentro de um programa. O número zero, colocado entre o **PRINT** e a mensagem a ser escrita, provoca o efeito desejado. Para limpar a parte de baixo da tela sem afetar a área normal de escrita, use a instrução:

```
20 INPUT ""
```

Procure não deixar um espaço em branco entre as aspas, pois o comando não funcionará.

Experimentando um pouco mais, você fará novas descobertas sobre a parte inferior da tela. Verá, por exemplo, que ela não é limitada às duas linhas antes mencionadas. Dessa maneira, se mais palavras forem incorporadas ao programa, a parte de baixo se expandirá, "empurrando" para cima a parte superior da tela.

Pode-se também usar o comando **PRINT AT** na parte inferior da tela, neste caso, porém, a numeração deve ser independente.

Tente o seguinte:

```
10 PRINT 0;"INPUT":AT 12,0;
   "APRESENTA DICAS"
```

Se você quiser limpar a parte de baixo, após uma pequena pausa, acrescente estas linhas:

```
20 PAUSE 0
30 INPUT ""·PAUSE 0
```

E, para obter sucessivas repetições, acrescente:

```
40 GOTO 10
```

Finalmente, se para você é difícil perceber onde começa cada área independente da tela, adicione ao programa a linha a seguir, e rode-o novamente. Como você verá, o comando **BORDER** coloca uma moldura de cor diferente na parte de cima da tela.

```
5 BORDER 2
```

APLICAÇÕES

Mas para que servem esses truques? Tudo depende de sua capacidade de colocar a imaginação para funcionar.

A escrita na parte inferior da tela poderia ser utilizada, entre outras coisas, para deixar fora da área principal as mensagens de prontidão para entrada de dados. Assim, ao realizar, por exemplo, uma animação gráfica, será possível colocar mensagens num ponto em que não atrapalhem o movimento nem sejam destruídas por ele.

Outra aplicação importante consiste na criação de "janelas": áreas independentes de texto ou de gráficos, muito usadas nos "pacotes" mais modernos para micros profissionais.

UMA TELA LISTRADA

Pode-se obter um outro efeito interessante de tela quando o computador está sem fazer nada — aguardando, por exemplo, que uma tecla seja pressionada (função **INKEY$**). Nesse tipo de situação, sofistique seu programa com a rotina apresentada a seguir. Ela indica o estado de espera, fazendo a moldura da tela exibir continuamente um listrado cambiante de cores:

```
1 GOTO 100
2 BORDER 1:BORDER 2·BORDER 4·
BORDER 5 BORDER 6:BORDER 7
PAUSE 1:IF INKEY$="" THEN GOTO
10
3 LET X$=INKEY$:RETURN
80 REM --------- --------
90 REM PROGRAMA PRINCIPAL
95 REM - -  -----------
100 GOSUB 10
110 PRINT X$·GOTO 100
```

A instrução da linha 2, **PAUSE 1**, sincroniza as mudanças nas cores da moldura de modo que ocorram simultaneamente ao início da varredura da tela. Assim, elas parecem estar estacionárias. Experimente tirar o **PAUSE 1** para ver o que acontece.

O número de comandos **BORDER** também desempenha uma função: comandos que estejam a mais ou a menos forçarão uma perda da sincronização.

Introduza a rotina da linha 2 no começo do programa principal, que deve ser colocado a partir da linha 100. A linha 1 serve apenas para pular até o início do programa. O truque é necessário, pois o **GOTO** existente na linha 2 provocará um grande retardo se a sub-rotina estiver no final da listagem. Como o interpretador BASIC procura a linha de destino a partir do início do programa, sempre que encontra um **GOTO**, a demora será maior se houver um longo programa antes da linha que contém a instrução de desvio.

UMA AGENDA ELETRÔNICA (3)

Apresentamos aqui a parte final do nosso programa de agenda e calendário. Tendo listado as últimas rotinas, você poderá, finalmente, explorar todas as possibilidades desse versátil programa.

Se você já tem as duas partes anteriores deste programa gravadas em disco ou fita, carregue-as na memória do micro e digite as linhas que faltam. O programa estará completo e você poderá começar a usá-lo para manter um registro de todos os seus compromissos financeiros, sociais ou de saúde.

É aconselhável que releia as instruções para a utilização do programa dadas nos dois primeiros artigos da série. Instruções adicionais sobre gravação e leitura de dados da fita ou disco são fornecidas a seguir.

É fácil lidar com o programa. O menu principal permite que você escolha exatamente o que deseja, em cada situação. A todo instante, você encontrará uma mensagem indicando como proceder. Todos os pontos de entrada de dados têm uma rotina de verificação. Portanto, não se preocupe muito em digitar dados incompatíveis — o programa simplesmente voltará a solicitar o dado.

GRAVAÇÃO DOS DADOS

Após entrar os dados de sua agenda, escolha a opção 8 para gravá-los. No Apple II, o programa prevê o emprego de um drive; nos demais micros, de fita cassete. Há uma rotina especial para quem utiliza o TRS-Color com drives. Se você usa fita, deixe o gravador preparado para receber os dados quando solicitar a opção 8.

Ao usar novamente o programa, responda **S** à pergunta "Você tem listas de dados?". Os dados preexistentes serão carregados para a memória, podendo ser atualizados, apagados, listados. Antes de teclar **S**, deixe o gravador pronto para ler a fita com os dados.

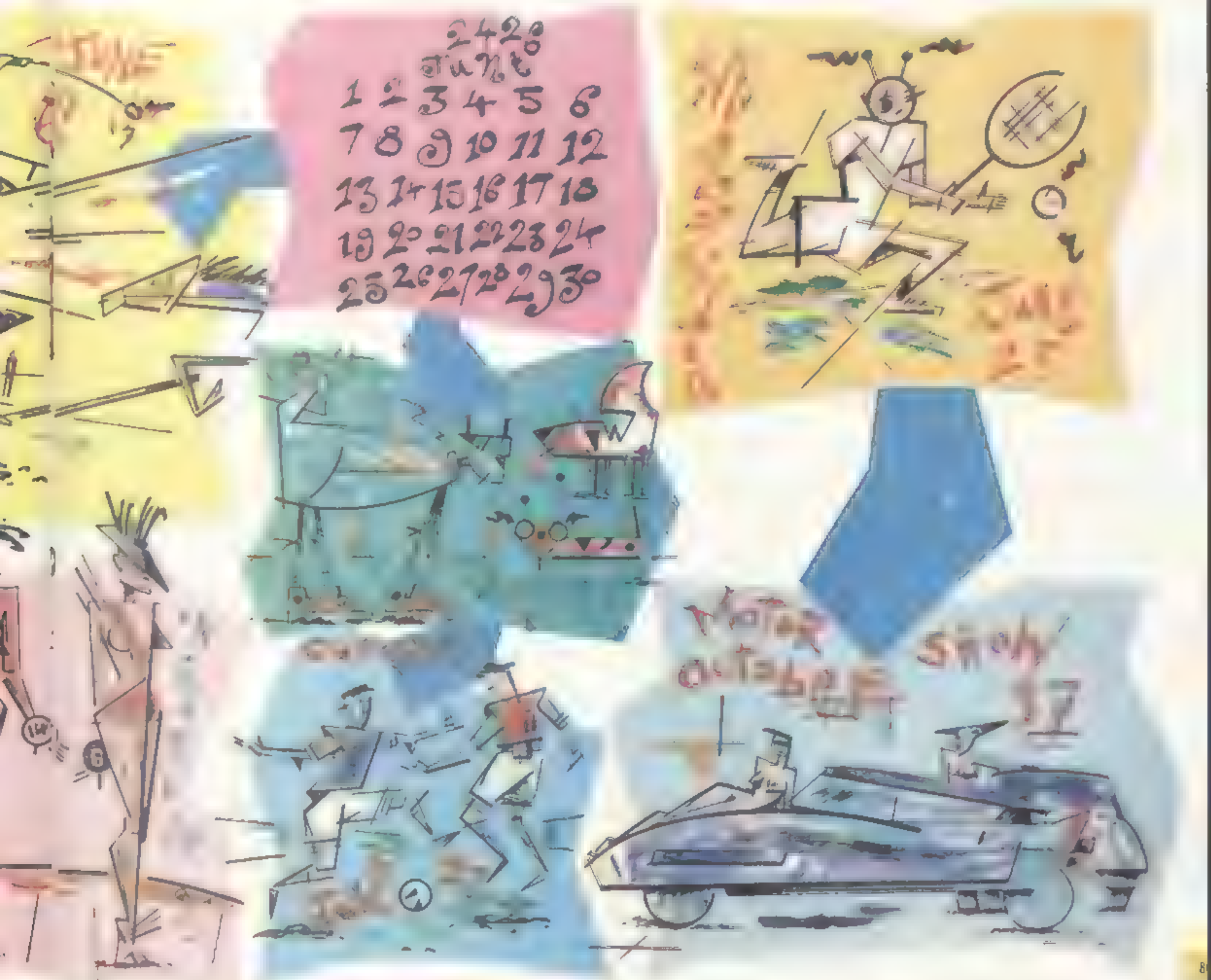
O nome para esse arquivo é "DIARIO". Assim, adote outro nome para o programa (CALENDÁRIO, por exemplo).

S

```
1760 LET M4=0: LET A4=0
1770 INPUT "ANO:",YR: IF YR<175
3 OR YR>29999 THEN  GOTO 1770
1780 GOSUB 640
1790 GOSUB 2480
1800 CLS
1810 POKE 23692,255
1820 PRINT #P;"ANO ";YR
1830 PRINT #P: LET KB=0: GOSUB
1920: PRINT #P
1840 GOSUB 2460
1850 FOR z=1 TO 12
1860 LET MO=z
1870 PRINT #P;M$(MO*9-8 TO MO*9
)
1880 LET T2=5: LET S2=0: GOSUB
2020
1890 IF P=2 THEN  IF INKEY$=""
THEN  GOTO 1890
1900 NEXT z
1910 RETURN
1920 LET X2=0: LET C2=0: LET D2
=0
1930 IF KB=0 THEN  LET X2=7: PR
INT #P
1940 IF P=3 THEN  LET KB=KB+1
1950 PRINT #P;Z$( TO X2);
1960 FOR d=1 TO 7
1970 INK 4: IF d=1 THEN  INK 2
1980 PRINT #P;Z$( TO KB);a$((d
1)*3+1 TO (d-1)*3+3);
1990 NEXT d
2000 INK 7
2010 RETURN
2020 PRINT
2030 IF P=3 THEN  PRINT AT 10,4
; FLASH 1;"SAIDA PARA A IMPRESS
ORA"
2040 LET M5=0: LET XP=0: LET X2
=0: LET W2=0: LET A$="": LET D$
=""
2050 IF S2=1 THEN  LET A$ " "
LET W2=4
2060 IF S2=0 THEN  LET X2=7  LE
T W2 3
2070 IF P 3 THEN  LET A$=A$+" "
: LET W2 W2+1
2080 LET DA=1
2090 LET KB=MO: GOSUB 270: LET
M5 KB
2100 GOSUB 560: LET K2=7: LET X
P FN M(KB)
2110 PRINT #P,Z$( TO XP*W2);
2120 LET DA=0
2130 PRINT #P;Z$( TO X2);
2140 LET DA=DA+1: LET D$=A$+(ST
R$ (DA))+" ": IF LEN D$<W2 THEN
  LET D$=D$+Z$( TO W2-LEN D$)
2150 IF A$="" THEN  PRINT #P;D$
;: GOTO 2170
2160 LET KB=T2: GOSUB 350: PRIN
T #P; INK KB;D$;
2170 LET XP=XP+1
2180 IF NOT (XP>6 OR DA=M5) THE
N  GOTO 2140
2190 LET XP=0: PRINT #P: IF S2=
1 THEN  PRINT #P
2200 IF DA<>M5 THEN  GOTO 2130
2210 IF MO=ME THEN  PRINT #P:
PRINT #P;"Domingo de Pascoa ";M
$(ME*9 8 TO ME*9);DE
2220 IF P=3 THEN  PRINT AT 10,0
;Z$: PRINT AT 10,13;"PRONTO"
```



```
2230 RETURN
2240 GOSUB 2510
2250 LET T2=0: LET MX=0: LET N2
=0: LET A$="": LET CL=0: LET M9
=MO: LET Y9=YR
2260 GOSUB 2480: CLS
2270 GOSUB 2570: PRINT #P
2280 PRINT #P;"DIA      ENTRADA"
: PRINT AT 0,18;"Qualquer tecla
 para"'TAB 18;"proxima entrada"
2290 PRINT #P
2300 GOSUB 2460
2310 IF MO=ME THEN  PRINT #P; I
NK 5,DE;"  Domingo de Pascoa"
2320 PRINT #P
2330 FOR t=1 TO 4
2340 LET MX=Q(t)
2350 IF MX=0 THEN  GOTO 2410
2360 FOR N=1 TO MX
2370 LET K$=L$(t,N)
2380 LET KB=t
```



```
2390 LET K2=3· GOSUB 470: IF K2
=1 THEN  PRINT #P; INK Z(t);K$(
2 TO 3);"      ";K$(10 TO )
2400 NEXT N
2410 IF INKEY$="" THEN  GOTO 24
10
2420 NEXT t
2430 FOR 1=1 TO 100· NEXT 1
2440 IF INKEY$="" THEN  GOTO 24
40
2450 LET MO=M9  LET YR=Y9· RETU
RN
2460 IF INKEY$="" THEN  GOTO 24
60
2470 RETURN
2480 PRINT · PRINT "DESEJA IMPR
IMIR (S/N) ?"· LET K$="sn"· GOS
UB 1480
2490 LET P=2: IF KB=1 THEN  LET
 P=3
2500 RETURN
2510 INPUT "MES ?";MO
```

MICRO DICAS

APLICAÇÕES PROFISSIONAIS DO PROGRAMA DE AGENDA

O programa de agenda e calendário, além de funcionar bem para a marcação e acompanhamento de compromissos e atividades de natureza pessoal, pode ter diversas aplicações profissionais tal como a marcação de consultas em uma clínica médica ou odontológica. Para isso será necessário modificar o programa em alguns pontos, pois o nível e o tipo de informações que uma agenda profissional requer são bem diferentes dos de uma agenda pessoal.

Tomemos como exemplo a agenda profissional de um dentista. Além de informações como data, horário da consulta, nome, sexo e idade do paciente, ela pode incorporar outros tipos de dado — entre eles, o procedimento a ser adotado (obturação, cirurgia, colocação de prótese etc.), a duração prevista para a consulta, a sala em que ela será realizada. Outros dados podem ser incluídos, como o saldo ou débito financeiro do paciente e a fonte pagadora (convenio, por exemplo).

Os relatórios obtidos a partir de uma agenda também são muito diferentes para um profissional. No caso do dentista, ele poderia precisar não só da agenda diária impressa, com a lista dos pacientes a serem atendidos e demais dados mencionados, mas, ainda, da listagem dos pacientes que faltaram, de uma análise estatística dos procedimentos realizados no mês etc.

```
2520 IF MO<1 OR MO>12 THEN  GOT
O 2510
2530 INPUT "ANO ?";YR
2540 IF YR<1735 OR YR>29999 THE
N  GOTO 2530
2550 GOSUB 640
2560 RETURN
2570 PRINT #P; PAPER 1; INK 7;M
$(MO*9-8 TO MO*9);" ";YR
2580 RETURN
```



```
2240 'IMPRIME MES -T2-S2
2250 M5=0:XP=0·X2=0·W2=0:A2$=""
:D2$=""
2260 IF S2=1 THEN A2$=" "·W2=4
2270 IF S2=0 THEN X2=7:W2=3
2280 IF P=2 THEN A2$=A2$+" ":W2
=W2+1
2290 DA=1
2300 KB=MO GOSUB 230:M5=KB
2310 GOSUB 560·K2=7:XP=FNM(KB)
2320 PRINT #-P,STRING$(XP*(W2),
32);
2330 DA=0
2340 REM
2350 PRINT #-P,STRING$(X2," ");
2360 REM
2370 DA=DA+1:D2$=A2$+MID$(STR$(
DA),2)+" ":IF LEN (D2$)<W2 THEN
 D2$=D2$+" "
2380 IF A2$="" THEN PRINT #-P,D
2$:GOTO 2400
2390 KB=T2 GOSUB 310·MID$(D2$,1
,1)=CHR$(KB)·PRINT#-P,D2$;
2400 XP=XP+1
2410 IF NOT(XP>6 OR DA=M5) THEN
 2360
2420 XP=0:PRINT #-P:IF S2=1 THE
N PRINT#-P
2430 IF DA<>M5 THEN 2340
2440 IF MO=ME THEN PRINT #-P·PR
INT#-P,"DOMINGO DE PASCOA ";MID
$(MN$,ME*9-8,9);DE
2450 RETURN
2460 'ROTINA DIARIO
2470 T2=0:MX=0:N2=0:A$="":CL=0:
M9=MO:Y9=YR
2480 GOSUB 2750:GOSUB 2720:CLS
2490 GOSUB 2820:PRINT #-P
2500 PRINT#-P,"DIA        ENTRADA"
2510 GOSUB 2660
2520 IF MO=ME THEN PRINT #-P,DE
;"DOMINGO DE PASCOA":PRINT #-P
2530 FOR T2=0 TO 3
2540 CL=159+16*T2
2550 IF P=2 THEN CL=32
2560 MX=VAL(LI$(T2,0))
2570 IF MX=0 THEN 2620
2580 FOR N2=1 TO MX
2590 A$=LI$(T2,N2)
2600 KB$=A$:GOSUB 470·IF K2=1 T
HEN PRINT #-P,MID$(STR$(ASC(MID
$(A$,2,1))),2);TAB(3);CHR$(CL);
RIGHT$(A$,LEN(A$)-4)
2610 NEXT:PRINT #-P
2620 IF P=0 AND INKEY$="" THEN
2620
2630 NEXT
2640 IF INKEY$="" THEN 2640
2650 MO=M9:YR=Y9:RETURN
2660 'ESPERA POR TECLA
2670 P1=PEEK(136):P2=PEEK(137)
2680 PRINT @480,"QUALQUER TECLA
 PARA CONTINUAR";
2690 IF INKEY$="" THEN 2690
2700 PRINT @480,STRING$(30,32);
·POKE 136,P1:POKE 137,P2
2710 RETURN
2720 'IMPRESSAO
2730 PRINT·PRINT"GOSTARIA DE IM
PRIMIR (S/N) ?":KB$="SN":GOSUB
1590:IF KB=1 THEN P=2
2740 RETURN
2750 REM
2760 INPUT"MES:";MO
2770 IF MO<1 OR MO>12 THEN 2760
2780 INPUT" ANO:";YR
2790 IF YR<1753 OR YR>29999 THE
N 2780
2800 GOSUB 650
2810 RETURN
2820 REM
2830 PRINT MID$(MN$,MO*9-8,9);"
 ";YR
2840 IF P=2 THEN PRINT #-2,MID$
(MN$,MO*9-8,9);"  ";YR
2850 RETURN
```



```
2240 ' imprime mês
2250 M5=0·XP=0·X2=0:W2=0·A2$=""
:D2$=""
2260 IF S2=1 THEN A2$=" ":W2=4
2270 IF S2=0 THEN X2=7:W2=3
2280 IF P=2 THEN A2$=A2$+" "·W2
=W2+1
2290 DA=1
2300 KB=MO·GOSUB 230·M5=KB
2310 GOSUB 560·K2=7·XP=FNM(KB)
2320 PRINT#P, STRING$(XP*W2,32)

2330 DA=0
2340 REM
2350 PRINT#P, STRING$(X2," ").
2360 REM
2370 DA=DA+1·D2$=A2$+MID$(STR$(
DA),2)+" ":IF LEN(D2$)<W2 THEN
D2$=D2$+" "
2380 IF A2$="" THEN PRINT#P,D2$
;·GOTO 2400
2390 KB=T2·GOSUB 310·MID$(D2$,1
,1)=CHR$(KB):PRINT#P,D2$;
2400 XP=XP+1
2410 IF NOT(XP>6 OR DA=M5) THEN
 2360
2420 XP=0 PRINT#P, IF S2=1 THEN
 PRINT#P,
2430 IF DA<>M5 THEN 2340
2440 IF MO=ME THEN PRINT#P,·PRI
NT#P,"Domingo de Páscoa    ",DE
;"de ";MID$(MN$,ME*9-8,9)
2450 RETURN
2460 'rotina diário
2470 T2=0·MX=0·N2=0·A$="" CL=0·
M9=MO·Y9=YR
2480 GOSUB 2750·GOSUB 2720·CLS
2490 GOSUB 2820·PRINT#P.
2500 PRINT#P,"Dia        Comprom
isso"
```

```
2510 GOSUB 2660
2520 IF MO=ME THEN PRINT#P,"dia
";DE,"Domingo de Páscoa":PRINT#
P,
2530 FOR T2=0 TO 3
2540 CL=32
2550 IF P=2 THEN CL=32
2560 MX=VAL(LI$(T2,0))
2570 IF MX=0 THEN 2620
2580 FOR N2=1 TO MX
2590 A$=LI$(T2,N2)
2600 KB$=A$:GOSUB 470:IF K2=1 T
HEN PRINT#P,MID$(STR$(ASC(MID$(
A$,2,1))),2);TAB(3);CHR$(CL);RI
GHT$(A$,LEN(A$)-4)
2610 NEXT:PRINT#P,
2620 IF P=3 AND INKEY$="" THEN
2620
2630 NEXT
2640 IF INKEY$="" THEN 2640
2650 MO=M9:YR=Y9:RETURN
2660 RETURN
2720 ' opção impressão
2730 PRINT:PRINT"Quer imprimir?
 (S/N)":KB$="SN":GOSUB 1590:IF
KB=1 THEN P=2
2740 RETURN
2750 ' rotina de data
2760 INPUT " Mês:";MO
2770 IF MO<1 OR MO>12 THEN 2760
2780 INPUT " Ano:";YR
2790 IF YR<1753 OR YR>29999 THE
N 2780
2800 GOSUB 650
2810 RETURN
2820 ' título
2830 PRINTMID$(MN$,MO*9-8,9);"
";YR
2840 IF P=2 THEN LPRINT MID$(MN
$,MO*9-8,9);"  ";YR
2850 RETURN
```



```
2240  REM  IMPRIME [illegible]
2250 M5 = 0:XP = 0:[illegible] = 0:W2 =
0:A2$ = "":D2$ = "[illegible]
2260  IF S2 = 1 THEN [illegible] = [illegible]":
W2 = 4
2270  IF S2 = 0 THEN X2 [illegible]:W2
= 3
2280  IF P = 2 THEN A2$ = A2$ +
" ":W2 = W2 + 1
2290 DA = 1
2300 KB = MO: GOSUB 230:M5 = KB

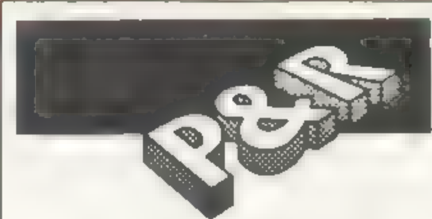
2310  GOSUB 560:K2 = 7:XP =  FN
 M(KB)
2320  PRINT  SPC( XP * W2)
2330 DA = 0
2340  REM
2350  PRINT  SPC( X2)
2360  REM
2370 DA [illegible] DA [illegible]D2$ = A2$ +  S
TR$ (DA[illegible] " ": IF  LEN (D2$) <
[illegible] THEN D2$ = D2$ + " "
[illegible]80  I[illegible] = "" THEN  PRINT D
[illegible]$;: GOTO [illegible]
[illegible]390 KB = T2 [illegible]OSUB 310:D2$ =
CHR$ (KB) + [illegible]D$ (D2$,2): PRIN
T D2$[illegible]
2400 [illegible] + 1
2410  IF  NOT (XP > 6 OR DA = M
5) THEN 2360
2420 XP = 0: PRINT : IF S2 = 1
THEN  PRINT
2430  IF DA <  > M5 THEN 2340
2440  IF MO = ME THEN  PRINT "D
OMINGO DE PASCOA:   ";DE;" DE "
; MID$ (MN$,ME * 9 - 8,9)
2450  RETURN
2460  REM  ROTINA DIARIO
2470 T2 = 0:N2 = 0:MX = 0:A$ =
"":CL = 0:M9 = MO:Y9 = YR
2480  GOSUB 2750: GOSUB 2720: H
OME
2490  PRINT D$;"PR#";P: GOSUB 2
820: PRINT
2500  PRINT "DIA       COMPROMIS
SO"
2510  GOSUB 2660
2520  IF MO = ME THEN  PRINT DE
;" - DOMINGO DE PASCOA"
2530  FOR T2 = 0 TO 3
2540 CL = 40
2550  IF P = 1 THEN CL = 32
2560 MX =  VAL (LI$(T2,0))
2570  IF MX = 0 THEN 2630
2580  FOR N2 = 1 TO MX
2590 A$ = LI$(T2,N2)
2600 KB$ = A$: GOSUB 470: IF K2
 = 1 THEN  PRINT  STR$ ( ASC (
MID$ (A$,2,1))); TAB( 3); CHR$
(32); RIGHT$ (A$, LEN (A$) - 4)

2610  NEXT : PRINT
2620  IF P = 0 AND T2 < 3 THEN
 GET R$
2630  NEXT : PRINT "<>":P = 0:
PRINT D$;"PR#";P: GET R$
2640 MO = M9:YR = Y9: RETURN
2660  RETURN
2720  REM  OPCAO IMPRESSAO
2730  PRINT : PRINT : PRINT "QU
ER UMA COPIA IMPRESSA? (S/N) ":
KB$ = "SN": GOSUB 1590: IF KB =
 1 THEN P = 1
2740  RETURN
2750  REM  ROTINA DE DADOS
2760  INPUT " MES:";MO
2770  IF MO < 1 O[illegible] THEN
 2760
2780  INPUT " ANO:";YR
2790  IF YR < 1753 OR YR [illegible] 2999
9 THEN 2780
2800  GOSUB 650
2810  RETURN
2820  REM  ROTINA TITULO
2830  PRINT  [illegible] (MN[illegible],MO * 9
 - 8,9);" ";YR
2840  REM
2850  RETURN
```



MUDANÇAS PARA DRIVE

Eis a rotina para adaptar o programa do TRS-Color a um drive: ela inclui apenas pequenas modificações nos comandos que lidam com o arquivo



A agenda é uma espécie de banco de dados?

Sim, de certa forma, uma agenda é um banco de dados. Cada linha, reservada para a anotação de um compromisso, corresponder a a um registro de banco de dados. As informações sobre datas, horários e as descrições seriam os campos básicos.

Tanto é assim que poderíamos utilizar o programa de banco de dados publicado nos artigos das páginas 68, 688 e 706 para estruturar uma agenda diária. Com esse objetivo, definiríamos os seguintes campos

1. DIA: dois caracteres, numérico
2. MÊS: dois caracteres, numérico
3. ANO: dois caracteres, numérico
4. HORÁRIO: quatro caracteres, numérico
5. ATIVIDADE: trinta caracteres, alfabético
6. TIPO: um caractere alfabético
7. STATUS um caractere alfabético

O campo TIPO serv r a para identificar a categoria de compromisso ou atividade (por exemplo, L para lazer, P para profissional, F para familiar e assim por diante) Já o campo STATUS teria a finalidade de informar se a atividade marcada foi realizada ou não

Depois de entrar seus compromissos utilizando esse esquema, será possível pesquisar o banco de dados por meio das funções de PESQUISAR E LISTAR do programa gerenciador de bancos de dados.

Com esse procedimento, você terá transformado o programa de banco de dados não em uma simples agenda, mas num poderoso e flexível instrumento de gestão pessoal

```
1750 OPEN"O",#1,"DIARIO/DAT"
1780 PRINT #1,LI$(N,0)
1810 FOR J=1 TO 4:PRINT#1,STR$(
ASC(MID$(LI$(N,P),J,1))):NEXT J
1820 PRINT #1,MID$(LI$(N,P),5)
1850 CLOSE #1
1890 OPEN"I",#1,"DIARIO/DAT"
1910 LINE INPUT #1,LI$(N,0)
1950 FOR J=1 TO 4:INPUT#1,NN$:L
I$(N,P)=LI$(N,P)+CHR$(VAL(NN$))
:NEXT J
1960 LINE INPUT #1,NN$:LI$(N,P)
=LI$(N,P)+NN$
1990 CLOSE [illegible]
```

O JOGO A RAPOSA E OS GANSOS (1)

Era uma vez uma raposa que jogava xadrez contra um grupo de gansos e... Bem, se você quer saber o fim dessa história, leia o artigo a seguir e entre no jogo.

Um programa como o do jogo chamado *Otelo* pode ser escrito — como vimos – de maneira que o computador dispute contra um adversário humano sem que este fique numa situação de inferioridade. Mas, neste caso, com um pouco de prática, pode-se, quase sempre, ganhar do computador.

Nas próximas três seções de *Programação de Jogos*, veremos um programa mais sofisticado escrito para que o computador dispute o jogo chamado *A Raposa e os Gansos*. Esse programa oferece vários níveis de dificuldade, permitindo que o jogo se torne tão fácil ou tão difícil quanto se queira.

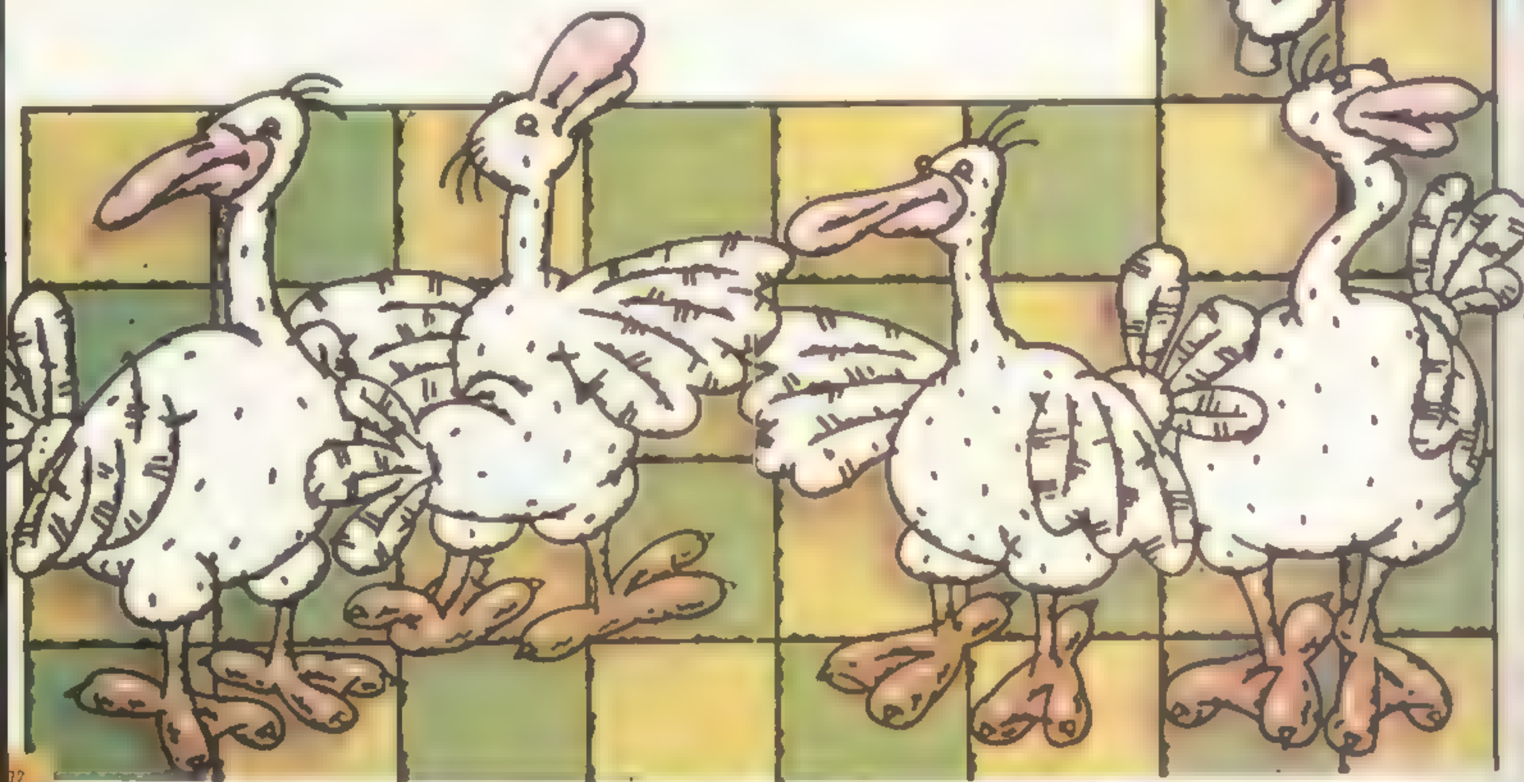
A Raposa e os Gansos serve para ilustrar muitos dos problemas e dos princípios envolvidos na elaboração de um dos mais interessantes e difíceis jogos que existem: o xadrez.

Para os programadores, contudo, esse jogo apresenta um inconveniente: é impossível escrever em BASIC um programa de xadrez que valha a pena, pois a máquina levaria longo tempo para executar cada jogada. Nos níveis superiores de *A Raposa e os Gansos*, o programa despende um bom tempo para analisar cada movimento, chegando mesmo a perder cerca de meia hora nos níveis mais altos e nas máquinas mais lentas. O problema pioraria muito se tentássemos escrever um jogo de xadrez.

Para manter nosso trabalho longe das dificuldades do código de máquina, precisamos de um jogo mais simples. *A Raposa e os Gansos* preenche esse requisito, porque, além de ter várias características do xadrez, é jogado no mesmo tipo de tabuleiro.

O artigo está dividido em três partes. Nesta primeira, você terá acesso aos princípios que fundamentam o programa e dará início à estruturação do jogo. Nas duas seguintes, construiremos os segmentos restantes.

Vamos começar analisando o jogo propriamente dito e quais as exigências necessárias para disputá-lo.



- FAÇA DO COMPUTADOR UM JOGADOR INTELIGENTE
- RAPOSA, GANSOS E XADREZ
- TEORIA DE PROGRAMAÇÃO
- EXPLICANDO O JOGO
- COMO RESOLVER PROBLEMAS DE PROGRAMAÇÃO
- COMO AVALIAR POSIÇÕES NO JOGO
- INICIE O PROGRAMA



O JOGO

A Raposa e os Gansos é jogado nas casas brancas (verdes, nas ilustrações) de um tabuleiro de xadrez. Nele, temos uma raposa que inicia a disputa em um lado do tabuleiro e quatro gansos que partem do lado oposto. Um jogador controla a raposa e outro, os gansos. Para a raposa, o objetivo do jogo é passar pelas posições das aves e atingir o lado oposto; para os gansos, tudo se resume em encurralar a raposa.

O jogo pode parecer desequilibrado, já que se trata de uma disputa de quatro contra um; entretanto, o programa limita as investidas dos gansos para a frente, enquanto dá à raposa inteira liberdade de movimentos para a frente e para trás. Além disso, ele foi escrito de tal maneira que o computador pode jogá-lo tanto com a raposa, como com os gansos ou mesmo sozinho.

COMO RESOLVER OS PROBLEMAS

A Raposa e os Gansos apresenta outro ponto de semelhança com o xadrez: aqui também o fator sorte não é decisivo. Na verdade, o resultado depende apenas da perícia dos adversários. Embora seja teoricamente possível para o computador aprender a jogar por tentativa e erro, como um ser humano, esta não é a melhor forma de resolver os problemas envolvidos no jogo — e, de qualquer modo, um programa assim concebido não caberia no seu micro.

Programar um jogo como *A Raposa e os Gansos* ou como o xadrez é um exercício de inteligência artificial. Para constituir um desafio a um jogador humano, o computador precisa jogar inteligentemente. Porém, a máquina não tem condições de olhar para o tabuleiro e captar as relações espaciais entre as peças, tal como nós o fazemos. Assim, as posições devem ser convertidas em valores numéricos que possam ser analisados pelo computador.

Para escrever um programa que permita ao computador jogar de modo inteligente, deve-se primeiro estudar a natureza do jogo. Esta determinará o tipo de programa. É importante, também, ter como referência um método ideal, inteiramente documentado. Prestam-se bem a esse tratamento a resolução do cubo mágico, aberturas de xadrez etc. Em comparação com tais exemplos, nossa tarefa é mais simples.

Quando o elemento sorte está presente, é possível usar uma estratégia de movimento único — ou seja, o programa analisa as alternativas de jogo apenas um movimento à frente para, então, escolher a melhor jogada. O ludo, por exemplo, seria um candidato a essa estratégia, exigindo rotinas de decisão relativamente simples.

Em jogos em que a sorte conta pouco ou nada, como em *A Raposa e os Gansos*, pode-se usar uma estratégia de movimentos consecutivos, onde se analisa uma série de passos adiante para investigar os resultados possíveis. Ao encontrar o movimento mais vantajoso, o programa o executa.

AVALIAÇÃO DA POSIÇÃO

Para que o computador possa decidir qual é o melhor movimento entre vários possíveis, atribui-se um valor numérico a cada quadrado do tabuleiro. No nosso jogo, é vantajoso para a raposa estar o mais à frente possível — lembre-se de que ela ganha o jogo ao atingir o lado oposto do tabuleiro.

As colunas são numeradas alternadamente da esquerda para a direita e da direita para a esquerda. Assim, como os maiores números se alternam à direita e à esquerda, a raposa tenderá a se mover em linha reta.

Mas há outro elemento que deve ser considerado na avaliação de posições. Os cinco quadrados brancos imediatamente à frente da raposa têm um significado especial para o jogo. Se você olhar para a figura 1, verá cinco quadrados marcados de A a E. Se houver apenas um ganso em um desses quadrados, a raposa ganhará; se os gansos forem dois, mas não estiverem nas posições A ou B, ou se eles forem três, posicionados em ACD, BDE, ACE ou BCE, a vitória também caberá à raposa.

No início de cada turno, quando o computador joga tanto pela raposa quanto pelos gansos, o programa vai para uma sub-rotina que examina as posições de todas as peças e transforma essa informação em um único número, que o computador utiliza para escolher a melhor jogada. Quando a decisão é tomada, o programa converte o número novamente em posições.

BUSCA EM ÁRVORE

Os movimentos possíveis de uma posição do tabuleiro podem ser representados por uma estrutura em "árvore", com ramificações a partir da posição da peça. As alternativas de movimento a partir da posição da peça da figura 2, por exemplo, estão ilustradas no desenho da figura 3. Se examinarmos um movimento adiante, a árvore ficará mais complicada, passando a incluir o segundo nível de alternativas, também indicado na figura 3.

Observe que nem sempre teremos quatro "ramos" na estrutura, visto que a peça pode estar bloqueada por outra ou posicionada em um dos quadrados que limitam o tabuleiro.

COMO ACELERAR O PROGRAMA

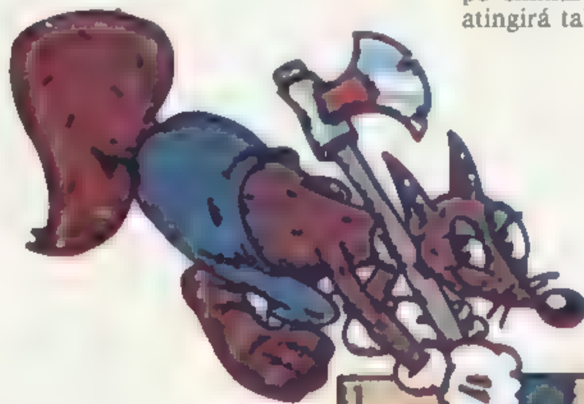
Devido ao grande número de operações que devem ser feitas, o BASIC pode se revelar demasiado lento. Mas, com o uso de determinados procedimentos, é possível acelerar o programa. Inicialmente, deve-se garantir que o programa não continue avaliando o próximo passo do adversário se o seu movimento já ganhou o jogo. Isso, no entanto, só economiza tempo no fim do jogo e não durante o seu desenrolar.

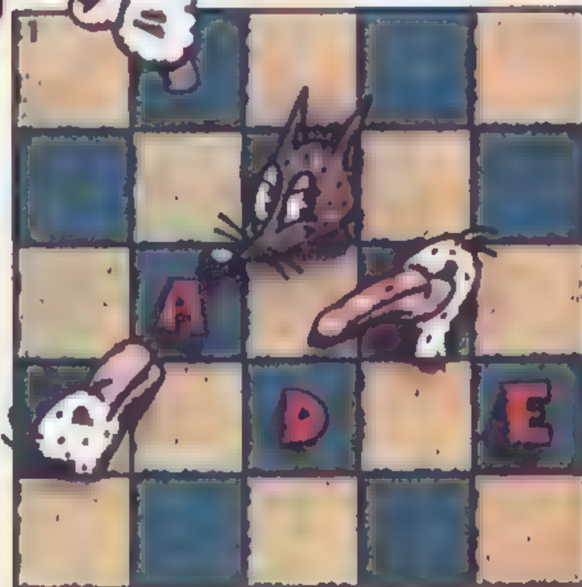
Em segundo lugar, é preciso levar em conta que uma mesma configuração de posições pode ser encontrada freqüentemente, a partir de jogadas diferentes. Convém, assim, elaborar uma tabela de configurações comuns com os valores já calculados associados a elas. Essa tabela fará com que o computador pare de analisar posições que se repetem. Tenha, porém, o cuidado de não armazenar dados de uma posição cuja avaliação pode ser mais rápida do que uma consulta à tabela Teoricamente, o emprego da tabela só é vantajoso nas situações em que o computador avalia três lances completos — ou seja, três jogadas de ambos os jogadores — ou mais. No entanto, a prática tem demonstrado que a armazenagem só vale realmente a pena quando cinco ou mais lances estiverem sendo analisados.

Finalmente, há a possibilidade de se utilizar o algoritmo alfa-beta, descoberto no início dos anos 60 por pesquisadores da área de inteligência artificial. Recorre-se a ele sempre que, numa busca em árvore, a análise envolve mais de um nível.

Voltemos à figura 3, que mostra os movimentos possíveis de uma posição da raposa. O programa vai avaliar todo o ramo A, depois o ramo B. O melhor movimento será armazenado pelo programa e comparado com os resultados de cada ramo. Ao encontrar, em qualquer ponto, um resultado pior do que um anterior, todo o ramo será rejeitado.

Quando a árvore se torna mais complexa, o algoritmo alfa-beta é realmente útil, podendo descartar cerca de 99,8% das possibilidades em um estágio bem inicial, com um ganho de tempo similar. Neste jogo, a árvore não atingirá tal nível de complexidade.





INICIE O PROGRAMA

Agora que você já tem uma noção da teoria envolvida num jogo como *A Raposa e os Gansos*, passe à digitação da primeira parte do programa. Ela se encarrega de montar alguns desenhos, mas você não os verá por enquanto, se executar o programa. Não se esqueça de gravar as linhas digitadas.

S

```
  30 DEF FN A(F)=INT (LN (F)/L2
 +.001)
 100 GOTO 2002
2002 GOSUB 5000
2006 BORDER 4· PAPER 4: CLS
2008 PRINT AT 8,7; FLASH 1:"AGU
ARDE A PREPARACAO"
5000 FOR J=USR "A" TO USR "D"+7
: READ A: POKE J,A: NEXT J· RET
URN
5070 DATA 20,28,55,127,15,20,40
,72,0,0,248,252,250,40,20,20,0,
0,0,7,205,123,60,15,12,20,31,15
2,248,216,48,224
6000 LET S$(1)="   1   2   3
4"
6010 LET S$(2)=" 8   7   6   5
 "
6015 LET S$(3)="   9  10   11
 12"
6030 LET S$(4)="16  15  14  13
 "
6040 LET S$(5)="  17  18  19  2
0"
6050 LET S$(6)="24  23  22  21
 "
6060 LET S$(7)="  25  26  27  2
8"
6070 LET S$(8)="32  31  30  29
 "
6100 RETURN
```



```
5 CLS:PRINT @230,"DEFININDO GRA
FICOS":GOSUB 4000
10 SCREEN 1,0:GOTO 10
4000 PMODE 3:PCLS:DIM GS(5),FX(
4),SQ(10)
4010 DRAW"BM3,0C2FGR3FR5E3RED3F
DGLGNFLNG2LH2LH":PAINT(12,5),2·
PSET(0,1,4):PSET(14,5,4):PSET(1
6,4,4):GET (0,0)-(19,9),GS,G
4020 DRAW"BM18,20C4GL13HLG2R2F2
ND4R10ND4U2":PAINT(10,22),4:PSE
T(2,21,1):GET(0,20)-(19,28),FX,
G
4030 LINE(0,0)-(175,175),PSET,B
F:COLOR 3:LINE(8,8)-(167,167),P
SET,BF
4040 FOR K=8 TO 128 STEP 40:FOR
 L=28 TO 148 STEP 40:PUT(L,K)-(
L+19,K+19),SQ,PSET:PUT(L-20,K+2
0)-(L-1,K+39),SQ,PSET·NEXT L,K
4050 PUT(68,13)-(87,21),FX,PSET
```

```
4060 FOR K=8 TO 128 STEP 40:PUT
(K,153)-(K+19,162),GS,PSET:NEXT
4070 THS="R2ND6R2BR4D6BR4U3LBR6
ND3BU2UBF3ND3R4D3BR5U6D3NR3F3BR
4U3BU2UBF3BR2ND3R4D3BR7L3U3R3D6
L3BE3BR4RBR5RBR5R"
4080 MWS="ND6F3RU3D6BR9L4U3R4D3
BE3F3UE2BR3R3DL3D2R3BR6NU6E2F2N
U6BR3U6D3R4D3BR7L3U3R3DL3BR7NUN
R2D2BR8L3U3R3DL3BE4BR7R4D3L4D3"
4090 WG$="RD6E2F2U6BR4D6BR4U3NL
2BF3U3BU2UBF3BRNR3D3R3BR4NU6BR4
U3NL3BE3BR8L4D6R4U3BR4D3R4U3L4B
R8D3R4U3L3BR9BUL2D2R2D2L2BR9L3U
3R3DL"
4100 VS="T402DEFBGACDEGGDCDE"
4200 C=1:G=0:RETURN
5000 FOR K=1 TO 14:PUT(200,5)-(
210,15),SQ,PRESET:PLAY"T50AC":P
UT(200,5)-(210,15),SQ,PSET:PLAY
"DA":NEXT:RETURN
```



```
5 CLS· COLOR 1,11,11:SCREEN0
7 KEYOFF:GOSUB 4000
10 GOTO 10
4000 SCREEN2,2·FX=1·A$=""·FORK=
2TO5·GS(K-1,0)=K·NEXT:FOR K=1TO
4:GS(K,1)=32-K:NEXT
4010 FOR K=1 TO 32:READ A·A$=A$
+CHR$(A):NEXT:FOR K=1 TO 4:SPRI
TES(GS(K,0))=A$·NEXT:A$=""
4020 FOR K=1 TO 32·READ A·A$=A$
+CHR$(A)·NEXT:SPRITES(FX)=A$.
4030 LINE (10,10)-(185,185),6,B
F·LINE (18,18)-(177,177),11,BF
4050 FOR K=18 TO 138 STEP 40:FO
R L=38 TO 158 STEP 40:LINE (L,K
)-(L+19,K+19),3,BF:LINE (L-20,K
+20)-(L-1,K+39),3,BF:NEXT:NEXT
4060 PUT SPRITE FX,(78,20),6
4070 FOR K=0 TO 3:PUT SPRITE GS
(K+1,0),(18+K*40,159),15:NEXT
4080 VS="T2000 3DEFBGACDEGGDCDE"
4100 DATA 0,0,12,60,12,6,6,7,7,
3,3,1,0,0,1,2,0,0,0,0,0,8,8,248
,248,248,248,240,224,192,32,16
4110 DATA 0,0,0,2,2,6,62,27,3,3
,2,2,2,2,0,0,0,0,0,0,0,4,8,248,
240,240,16,16,16,16,0,0
4200 RETURN
5000 BEEP:RETURN
```



```
2  LOMEM: 16384
5  HOME · GOSUB 4000: GOSUB 310

10  END


4000  DATA  02,00,06,00,42,00
4010  DATA  45,62,63,63,46,45,4
5,53,63,63,119,41,53,127,9,45,4
5,45,62,63,63,55,45,45,53,63,63
,14,45,62,55,55,119,73,40,60,0
4020  DATA  45,45,44,44,44,54,5
5,63,62,63,46,45,45,45,45,45,45
,44,44,172,146,194,27,63,63,63,
63,55,45,45,45,45,62,63,63,63,4
6,62,62,126,73,73,37,37,36,0
4030  FOR A = 768 TO 768 + 88·
READ E: POKE A,E: NEXT
4040  POKE 233,3: POKE 232,0
4050  RETURN
5000  FOR K = 1 TO 3: PRINT  CH
R$ (7);· NEXT · RETURN
```

O TABULEIRO



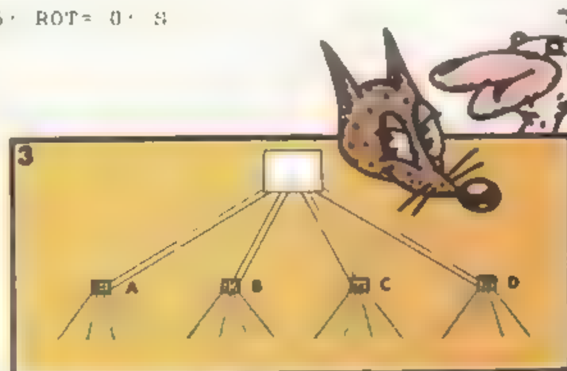
```
310 LET F=FN A(ABS (P))-30:
LET B=P/B(F)· IF B<0 THEN  LET
B=B+BX: LET F=33-F
320 LET C=B/B(29): FOR A=8 TO
1 STEP -1  LET R$(A)=B$(INT (C
)+1,(2-FN W(A))): LET C=(C-INT
 (C))*16: NEXT A
330 LET R$(INT (F/4+.8))(FN C(
F)+1 TO FN C(F)+4)=F$(FN W(F/4
-.2)+1)
340 FOR A=1 TO 8: PRINT AT 2*A
,8; PAPER 7;S$(A): PRINT AT 2*
A+1,8; PAPER 7;R$(A): NEXT A:
RETURN
```

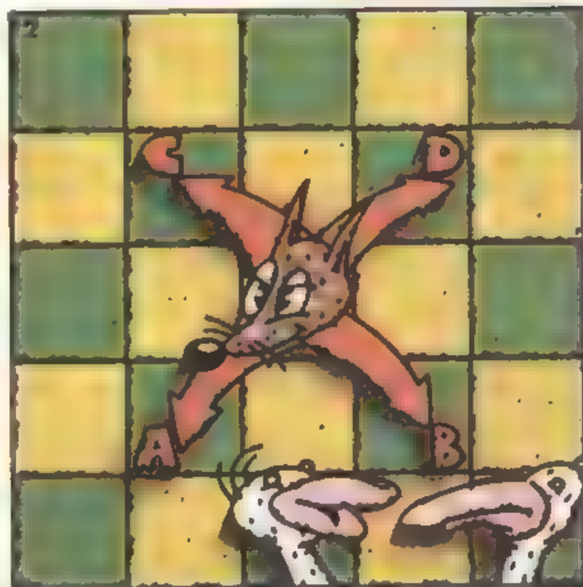


```
310  HGR · HCOLOR  5· ROT= 0· S
CALE= 1
320 X1 = 57·X2 = 219·Y1 = 0 Y2
= 159·GS = 1·FX = 2
330  HPLOT X1,Y1 TO X2,Y1 TO X2
,Y2 TO X1,Y2 TO X1,Y1
340  FOR I = 0 TO 7
350 IN = 20 * (I / 2 <  >  INT
(I / 2))
360  FOR J = 0 TO 19
370  FOR K = 0 TO 3
380  HPLOT 20 * I + J + X1 + 1,
(40 * K + IN) TO 20 * I + J + X
1 + 1,(40 * K + IN + 19)
390  NEXT · NEXT · NEXT · HCOLO
R= 3
395  DRAW FX AT X1 + 3 * 20 + 3
,Y1 + 8
400  FOR I = 0 TO 6 STEP 2· DRA
W GS AT X1 + I * 20 + 8,Y1 + 7
* 20 + 5: NEXT : RETURN
```

As linhas que vão de 310 a 350 mostram o tabuleiro com as cinco peças em posição. A sub-rotina é chamada uma vez a cada jogada, tanto pela raposa quanto pelos gansos.

As instruções para os demais micros fazem parte de uma outra sub-rotina de gráficos.





A ESCOLHA DA MEMÓRIA AUXILIAR

Existem dois tipos de memória auxiliar para micros: a fita cassete, barata e amplamente disponível, e o disquete, poderoso, mas bem mais caro. Veja aqui como fazer a melhor escolha.

Todos os computadores utilizam uma parte da memória central, chamada RAM (*Random Access Memory*), para armazenar programas e dados que são entrados pelo teclado ou carregados a partir de um periférico de armazenamento de dados. Esses dispositivos externos são necessários porque a informação armazenada na memória RAM desaparece quando o computador é desligado.

Existem diversos tipos de dispositivo de memória auxiliar destinados a microcomputadores. Os principais são as *fitas* e os *discos*. Ambos possibilitam o armazenamento permanente de informação, pois seu funcionamento é baseado no princípio da gravação magnética. Há, entretanto, uma grande diferença entre eles no que diz respeito ao custo de aquisição e à capacidade e velocidade de armazenamento.

Os discos, que constituem os dispositivos mais sofisticados e caros da memória auxiliar, dotam o computador pessoal de poderosos recursos, permitindo a programação de aplicações mais complexas, como bancos de dados de uso profissional e sistemas de controle para pequenas empresas.

ARMAZENAMENTO EM FITA

O periférico de armazenamento de dados mais utilizado em computadores pessoais é o gravador comum de fita cassete, que alia conveniência operacional e baixo custo. Para a maioria dos usuários domésticos, este é o único fator que conta.

Os gravadores de audiocassete existentes no mercado são, quase todos, bastante adequados para essa finalidade. A vantagem adicional é que existe uma enorme quantidade de software de preço acessível disponível em fita cassete, para a maioria dos computadores pessoais (sobretudo na área de jogos e entretenimento).

A tecnologia dos gravadores-cassete é muito simples. Seu uso com computadores requer apenas que um sinal audível seja gerado pela interface adequada para a gravação na fita. Assim, para que o sistema funcione a contento, basta montar uma conexão entre o computador e o gravador e ajustar corretamente os níveis de volume e tonalidade. Com alguns cuidados periódicos de manutenção, o usuário não terá problemas por longo tempo.

Bem, isto é o que acontece na teoria! Na prática, contudo, não é tão fácil conseguir a combinação ideal entre o periférico e o micro.

QUALIDADE DE GRAVAÇÃO

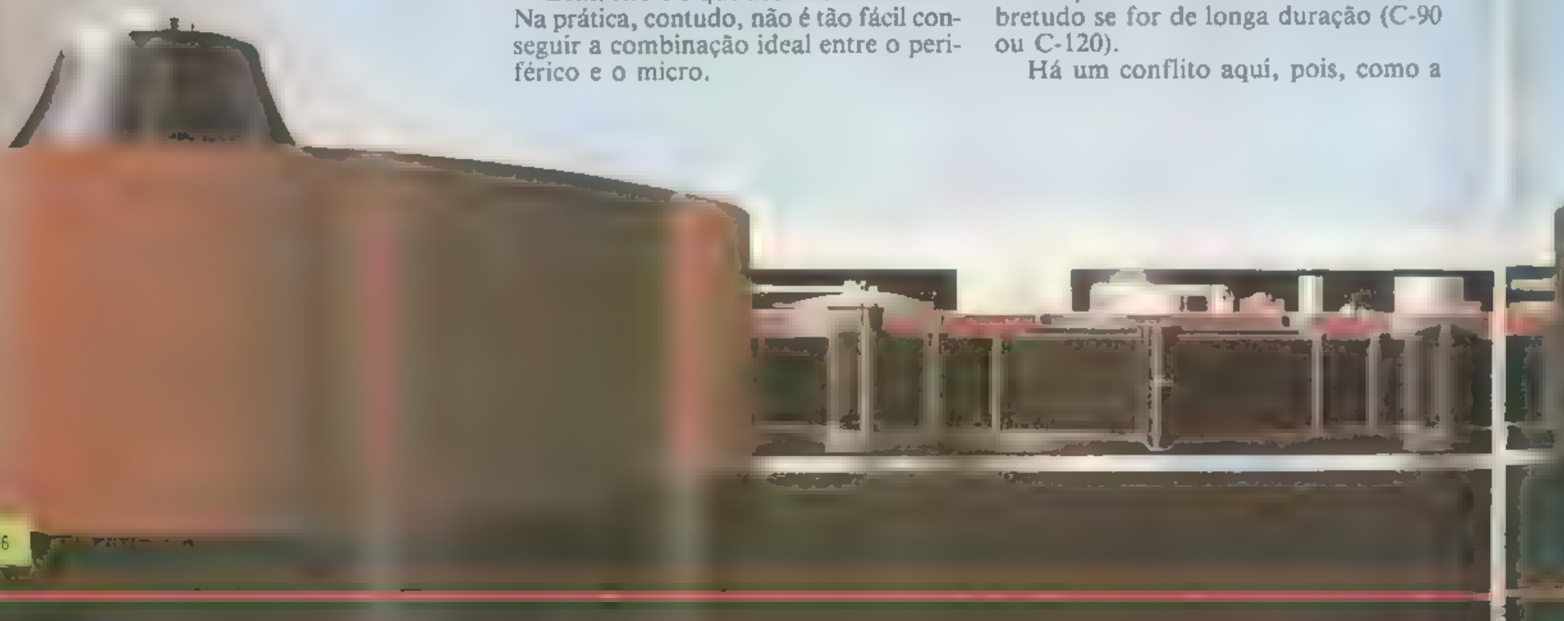
Uma boa maneira de contornar dificuldades mais sérias consiste em comprar um gravador apropriado para microcomputadores (*data recorder*). Eles diferem dos gravadores comuns por disporem de circuitos eletrônicos internos que processam sinais mais "limpos" de transmissão digital.

Gravadores especiais, como, por exemplo, os destinados a computadores da linha MSX, dão menos problemas na gravação e leitura de dados digitais e, também, possibilitam uma conexão mais simples com o computador. Nem esses aparelhos, porém, estão totalmente livres de falhas, pois é o sistema de gravação em fita, em si, que apresenta certas desvantagens técnicas.

O principal ao se usar fita é aumentar ao máximo a qualidade de gravação: esta é uma área, portanto, sobre a qual o usuário pode exercer algum controle. Convém lembrar que, mesmo quando a gravação foi bem feita, o sinal deteriora-se ao ser lido por um gravador de baixa qualidade. O resultado é pior ainda quando se tenta gravar alguma coisa com um aparelho que esteja em más condições.

A qualidade da fita também é muito importante. Basicamente, deve-se evitar a tentação de utilizar uma fita velha, sobretudo se for de longa duração (C-90 ou C-120).

Há um conflito aqui, pois, como a



- ARMAZENAMENTO EM FITA
- QUALIDADE DA GRAVAÇÃO
- TAXA DE TRANSMISSÃO
- ALÇAS DE FITA
- ACESSO SERIAL
- ACIONADORES DE DISCOS
- O SISTEMA OPERACIONAL DE DISCOS
- INTERFACES
- TIPOS DE SOFTWARE

capacidade da memória é limitada pelo método de gravação empregado, seria desejável ampliá-la, usando-se fitas de maior duração. Estas, porém, são bem mais finas do que as C-30 e, em consequência, podem sofrer alongamentos por efeito do vaivém contínuo e pela força relativamente grande exercida pelo rebobinamento e avanço rápidos.

O alongamento determina perda de bytes na fita e variações na amplitude do sinal. Ambos os efeitos são desastrosos, no que diz respeito às características de leitura por computadores: um byte perdido invalida a leitura de todo um programa, por exemplo. Na melhor das hipóteses, isto pode levar a carregamentos irregulares. Na pior das hipóteses, dados são perdidos durante a gravação!

Uma fita C-30 é a melhor opção prática: ela suporta quinze minutos de gravação de cada lado, o que é mais do que suficiente para armazenar programas muito extensos (até 64 Kbytes) na maioria dos computadores. Alguns micros das linhas Sinclair e TRS-80, entretanto, têm baixas velocidades de transmissão (300 baud), podendo exceder a capacidade da fita para dados e programas. Nesse caso, recomenda-se usar fitas C-60 de boa qualidade, mas sempre com muito cuidado.

Um programa de 32 Kbytes, por exemplo, gasta somente três a quatro minutos de fita nos micros das linhas TRS-Color e MSX, cerca de seis minutos nos micros da linha Spectrum, e quase quinze minutos no ZX-81. Faz mais sentido, portanto, recorrer a fitas de curta duração, gravando-se apenas um ou dois programas de cada lado, com uma duplicata, para fins de segurança. Esse procedimento não só favorece a conservação da fita como também torna muito mais fácil verificar onde está o início de uma gravação.

Existem fitas cassete de alta qualidade, feitas especialmente para gravação de dados digitais. Elas são normalmente disponíveis em três durações: C10, C15 e C20. É sempre bom testá-las antes de comprá-las em grande quantidade, pois são mais caras do que as fitas comuns.

TAXA DE TRANSMISSÃO

Os problemas com fitas cassete ficam bem mais sérios quando a taxa de transmissão de dados entre gravador e computador é mais alta. A taxa de transmissão, ou seja, a velocidade com que os bytes passam do computador para o gravador e vice-versa, é medida em bauds. Um baud, a grosso modo, corresponde ao número de bits transmitidos por segundo (na realidade, o número é calculado com base em um esquema um pouco mais complicado).

Quanto mais alta for a taxa de transmissão utilizada pelo computador, melhor deverá ser a qualidade do gravador e da fita cassete. Uma fita pode funcionar perfeitamente bem com um ZX-81 ou TRS-80 Modelo I, mas falhará se usada com um TK-85 com função *speed*, ou com um MSX. O mesmo problema ocorrerá até com computadores mais lentos, em determinadas situações — por exemplo, na gravação de jogos com rotinas próprias de carregamento rápido.

ALÇAS DE FITA

Para evitar os transtornos causados por gravadores convencionais de fita cassete, alguns fabricantes desenvolveram um sistema diferente: o cassete de fita sem fim (alça de fita). Os mais conhecidos são o *Microdrive* e o *Wafadri*



ve, para os micros da linha Sinclair Spectrum, mas existem diversas unidades disponíveis para computadores de outras marcas, usando interfaces padronizadas, como o RS-232C. Dispositivos desse tipo, infelizmente, não são encontrados no Brasil.

O acionador de alça de fita (*tape loop*) é um gravador que usa um cassete especial, com uma bobina sem fim (em alça). A fita desloca-se apenas em uma direção, mas muito rapidamente, de maneira que o ponto de início de um programa é atingido sem que seja necessário rebobinar a fita. Além de economizar tempo, o processo também protege a fita do desgaste provocado pelo avanço e rebobinamento rápidos dos gravadores convencionais.

Um Microdrive, por exemplo, com sua velocidade e capacidade (cerca de 100 Kbytes), permite que se eliminem todos os problemas das fitas convencionais. Seu desempenho, em alguns casos, aproxima-se do de acionadores de disquetes, muito maiores e mais caros.

Há algumas desvantagens, entretanto, nesse tipo de dispositivo: os cartuchos de fita são bem mais caros, a oferta de software no formato adequado é menor e, mesmo com maior velocidade e capacidade, ele não se presta para muitas aplicações nos campos educacional e profissional.

ACESSO SERIAL

Em qualquer sistema que utiliza fita, a transferência de dados entre gravador e computador é realizada sob a forma de uma sequência contínua de bytes, ou, em outras palavras, como uma fila indiana, do primeiro ao último byte. Para levar os dados ou programas de volta à memória, é preciso achar o ponto de início, e carregar a partir daí. Esse tipo de processo é chamado de *acesso serial*.

Para aplicações mais simples, como jogos ou outros programas pequenos, o método não apresenta desvantagens sérias, pois o programa precisa ser localizado e carregado apenas uma vez. Entretanto, oferece inconvenientes para aplicações em que, com freqüência, o computador precisa carregar dados ou segmentos de programa, em ordem diferente daquela em que estão gravados na fita. É o que ocorre com as aplicações de bancos de dados. Um dispositivo de alça de fita pode, quando o volume de dados é pequeno, "simular" um disquete, através da marcação de "blocos" fixos de dados na fita e da localização rápida dos mesmos por um software especial. Porém, havendo um grande volume de dados, a única solução é o acionador de disquetes.

ACIONADORES DE DISCOS

Os acionadores de discos são os dispositivos mais rápidos de memória auxiliar para microcomputadores. O tempo médio necessário para a localização de blocos de dados no disquete é medido, muitas vezes, em milésimos de segundo. A rapidez da gravação e a leitura de programas torna o dispositivo atraente principalmente para quem não consegue suportar a lentidão do **CSAVE** e **CLOAD** dos gravadores.

Os discos magnéticos são, além disso, muito mais confiáveis e seguros do que as fitas, desde que se tomem algumas precauções elementares. A perda de bytes por falhas no sistema de leitura ou gravação é rara e, quando ocorre, nem sempre o arquivo inteiro ou o disco ficam inutilizados.

Entretanto, o uso de discos impõe uma nova série de tarefas e procedimentos básicos, que exigem um grau de meticulosidade e precisão desnecessário com as fitas. Se você não tomar cuidado e utilizar inadequadamente os comandos de manipulação de arquivos em discos, disponíveis nos sistemas operacionais, as conseqüências poderão ser desastrosas. Existe, por exemplo, um comando que apaga um disco inteiro, em um piscar de olhos.

A fantástica capacidade de armazenamento dos discos magnéticos é uma forte razão para que se redobrem as precauções. Existem modelos com 10 a 20 megabytes de capacidade total, ou seja, aptos a armazenar até vinte milhões de caracteres (o que equivale, mais ou menos, à quantidade de texto de uma enciclopédia de trinta volumes!). Mais comumente, entretanto, a capacidade dos disquetes disponíveis para computadores pessoais é bem mais modesta: algo entre 150 a 360 Kbytes. Mas, mesmo assim, corresponde a uma grande quantidade de dados valiosos.

Infelizmente, os disquetes são vulneráveis a dobras, a arranhões e a agressões físicas do ambiente, tais como umidade e sujeira. Além disso, são sensíveis a campos magnéticos causados por alto-falantes, aparelhos de TV, motores de geladeira etc.

Para armazenamento de longa duração, a fita constitui um candidato bem melhor do que o disco. Assim, grandes empresas costumam manter seus arquivos vitais em bobinas de fitas, confiando aos discos apenas as operações diárias, mais rápidas.



Por todas as razões expostas, recomenda-se aos usuários de discos que façam, regularmente, cópias de segurança (*back-up*) dos arquivos de dados e de programas. A freqüência dessa operação depende muito da quantidade de informações geradas ou alteradas.

COMO FUNCIONA UM ACIONADOR

O funcionamento de um acionador de discos não é muito diferente do funcionamento de um gravador de fitas. O



O disco é formatado antes de ser usado, ou seja, é dividido em trilhas e setores. Cada setor tem a capacidade de 256 bytes.

sistema de leitura e gravação trabalha sobre um disco, chato, recoberto de material ferromagnético, como a fita, só que obedecendo a padrões de qualidade muito mais rigorosos.

Como o gravador, o acionador de discos possui uma cabeça de leitura e gravação, que é mantida em contato com a superfície magnética. A diferença é que a cabeça também pode ser movimentada, pulando de trilha para trilha da mesma maneira que o braço de um fonógrafo. Assim, com o disco rolando em alta velocidade, atinge-se rapidamente qualquer um de seus pontos.

Antes de sua utilização, o disco é *formatado*, ou seja, sua área total é dividida em *trilhas* concêntricas e estas, por sua vez, em segmentos iguais, chamados de setores. Os setores se subdividem em *blocos*, cada qual capaz de armazenar entre 128 a 256 bytes de informação. Esses blocos podem ser imaginados como se fossem as "gavetas" de um escaninho.

Um programa chamado *sistema operacional de discos* (DOS) possibilita a gravação ou leitura de informações nos blocos (cada um tem um endereço distinto, como na RAM). O DOS é um requisito essencial para a operação de um acionador de discos.

A cabeça de leitura e gravação pode ser direcionada pelo DOS, de modo a se posicionar sobre qualquer setor na superfície do disco, com uma velocidade bastante grande. Por isso, o disco magnético é um dispositivo de acesso arbitrário, muitas vezes mais rápido do que o acesso serial ou seqüencial da fita. Ocorre que ele também é um dispositivo de acesso serial, só que limitado a um setor por vez. Isso torna o sistema muito mais veloz, flexível e poderoso do que o baseado em fita.

TIPOS DE DISCO

Os discos magnéticos para computadores dividem-se, basicamente, em dois tipos: os *discos rígidos* e os *discos flexíveis*.

O disco rígido, de preço bem mais elevado do que o flexível, é confeccionado em metal, e, em geral, não é intercambiável, ou seja, permanece sempre dentro do acionador. A cabeça não encosta em sua superfície, mas paira a alguns milésimos de milímetro da mesma.

Graças a essa característica, o desgaste desse disco é nulo, e a densidade da gravação, muito maior. Atualmente, os discos rígidos para micro — também chamados discos *Winchester* — têm capacidades entre 5 a 40 Mbytes, dependendo do modelo.

O disco flexível, genericamente denominado disquete (ou *floppy*), é feito de plástico e protegido dentro de um envelope semi-rígido, lubrificado internamente. Pode ser adquirido em três tamanhos: 8 polegadas (19 cm), 5,25 po-

legadas (12,5 cm) ou 3,5 polegadas (8,5 cm) de diâmetro. O mais utilizado em computadores pessoais ainda é o de 5,25 polegadas, chamado de minidisquete, embora os microdisquetes (3,5 polegadas) estejam se tornando cada vez mais populares. Estes são embalados em cartuchos rígidos, o que lhes confere vantagens marcantes quanto ao manuseio e segurança. Sua capacidade é também bem maior, devido à superioridade na densidade de gravação.

A velocidade de rotação de um disco, seu tamanho, tipo de superfície e a presença ou não de uma segunda cabeça do outro lado do disco condicionam a capacidade e velocidade de acesso dos diversos modelos existentes. O usuário deve, portanto, levar em conta todos esses fatores, se pretende escolher o acionador de discos mais adequado às suas necessidades.

Os discos magnéticos classificam-se, ainda, em duas outras categorias: a categoria dos que só podem ser gravados em um lado (*discos de face simples*) e a dos que permitem a gravação nos dois lados, simultaneamente (*discos de face dupla*).

A *densidade de gravação* também diferencia os discos. Os mais baratos e de menor qualidade têm *densidade simples*: sua capacidade total e bastante restrita. Os de maior capacidade e velocidade de transmissão têm *densidade dupla*. Finalmente, existem os discos de *densidade quádrupla*, com o dobro do número de trilhas dos outros discos. Estes utilizam uma técnica de pulsos de gravação diferente da técnica dos demais e oferecem o grau máximo de capacidade e velocidade. Naturalmente, o preço do disquete aumenta de acordo com sua capacidade.

Para maior segurança, devem ser usados os disquetes com a especificação correta para cada densidade. Mas muitos usuários que trabalham com acionadores de face simples (linha Apple, por exemplo) costumam utilizar também o outro lado disponível no disquete, fazendo um pequeno orifício quadrado na margem não perfurada do envelope. Tal procedimento deve ser evitado no caso de discos especificados para densidade simples. Além disso, convém levar em conta que detritos oriundos da perfuração podem danificar o disco e, até mesmo, a cabeça do acionador.

SISTEMAS OPERACIONAIS

O controle dos acionadores de discos exige um software especial. Este é fornecido junto com o acionador, formando com ele uma unidade inseparável na utilização por um computador. Essencialmente um sistema operacional, o software gerencia todos os processos de localização de informação e de transferência entre computador e disco.

Em geral, o sistema operacional de disco (DOS) não reside na memória ROM permanente do computador, precisando ser carregado na RAM a partir do próprio disco, antes de ser ativado. Assim, uma parte da RAM é utilizada, ficando menos disponível para os programas e dados do usuário.

Alguns computadores pessoais podem usar mais de um sistema operacional com o mesmo acionador de disquetes. Certos sistemas operacionais, por outro lado, têm mais comandos ou, ainda, são mais poderosos e flexíveis do que outros. Mas é preciso cautela na escolha, pois a formatação imposta ao disco muitas vezes varia entre sistemas operacionais. Em consequência, o usuário não conseguirá utilizar programas feitos para um DOS, se tiver carregado um DOS diferente no computador. Na maioria das vezes, porém, não há sequer a possibilidade de escolha entre vários tipos de DOS, pois só existe um para aquela marca de computador.

INTERFACES E CONEXÕES

Se você tem a intenção de adquirir um ou mais acionadores de discos, considere cuidadosamente se a unidade em vista é realmente compatível com sua máquina. Além de um DOS adequado, é necessário também escolher um disco com interface que se adapte a seu sistema. Consulte os fabricantes — do computador e da unidade de discos — e procure aconselhar-se junto a colegas e conhecidos que já tenham tido sucesso na compra de uma unidade de disco para um computador igual ao seu.

SOFTWARE PARA DISCO

Comprar um acionador de discos pode ser muito bom para incrementar seu sistema, mas, para uma boa escolha, convém ter claro o que você pretende fazer com ele. Quer esteja pensando em desenvolver seus próprios programas — em Assembler, BASIC, PASCAL ou outra linguagem de programação —, quer pretenda apenas utilizar "pacotes" prontos de software, verifique primeiro se as ferramentas de programação ou os próprios programas estão disponíveis em quantidade e qualidade para o sistema que você tem em mente.

Sistemas baseados em microdisquetes, por exemplo, são recentes no mercado, mesmo em termos internacionais, e ainda existe muito pouco software para esse formato. Também é pequena a oferta de software próprio para sistemas de disquetes para a linha MSX. À medida que um sistema vai se difundindo, porém, essa situação tende a mudar. Mas micros implantados há mais tempo no mercado — como o Apple e o TRS-80 — terão sempre maior variedade de software que os demais.

Existe ainda a alternativa de se transferir software de um formato para outro. Certos programas comercialmente disponíveis, por exemplo, tornam possível copiar software de fita para disco; paralelamente, alguns sistemas operacionais reconhecem discos de diferentes densidades, e assim por diante. Em um próximo artigo trataremos em detalhes desse assunto.

COMO ESCOLHER?

O custo é, certamente, um fator bastante significativo no momento de decidir entre um sistema baseado em fita e outro baseado em disco. Muitas das vantagens dos discos não são importantes para usuários domésticos interessados apenas em aplicações como jogos. Para a maioria dos computadores pessoais não existem, em disquete, muitas opções de programas recreativos ou de videogames, e o pouco que se encontra custa mais caro do que a mesma versão para fita. Em contrapartida, há a possibilidade de que a versão para disquete seja mais sofisticada.

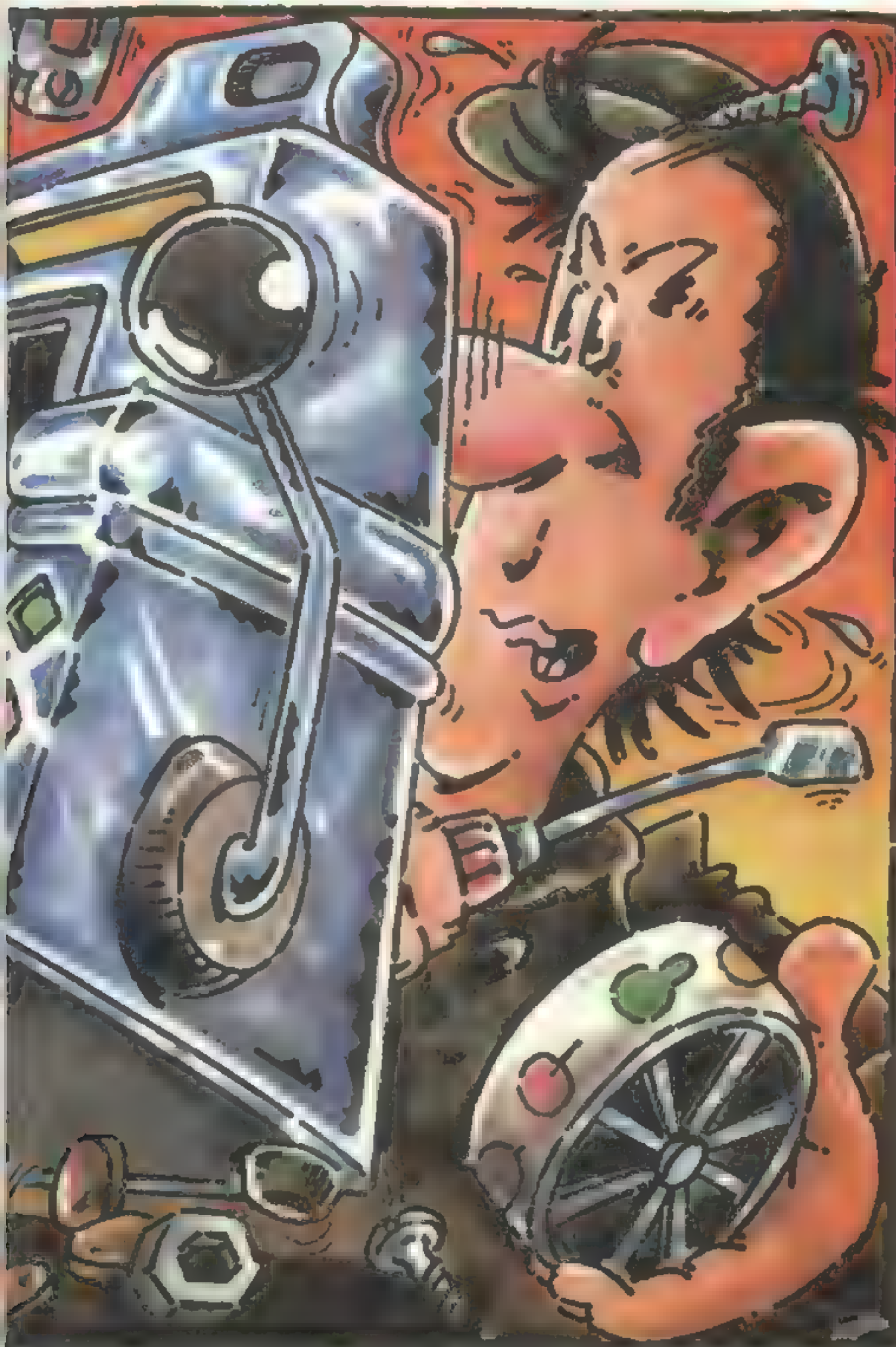
As unidades de disco são muito mais caras do que as de fita, custando mais do que o próprio computador. Às vezes, o preço não inclui o cabo de ligação, que também é caro.

É preciso considerar, ainda, o custo do suporte de informação. O preço de um disquete é muitíssimo mais elevado do que o de uma fita cassete de boa qualidade. Normalmente um disquete de densidade quádrupla custa quatro ou cinco vezes mais do que um de densidade simples. No entanto, deve-se levar em conta que, ao contrário da fita, o disquete permite aproveitamento *total* de sua capacidade.

Já os discos rígidos são tremendamente caros e úteis apenas em aplicações mais profissionais. É claro que seria muito bom ter todo o seu software disponível instantaneamente no mesmo disco, e carregá-lo em um piscar de olhos. Mas será que isso vale o preço de um Winchester?

O BANDIDO DE UM BRAÇO SÓ (2)

- COMPLETE SEU JOGO CAÇA-NÍQUEIS
- A ROTINA PRINCIPAL
- GIRE AS RODINHAS
- OS RESULTADOS PREMIADOS



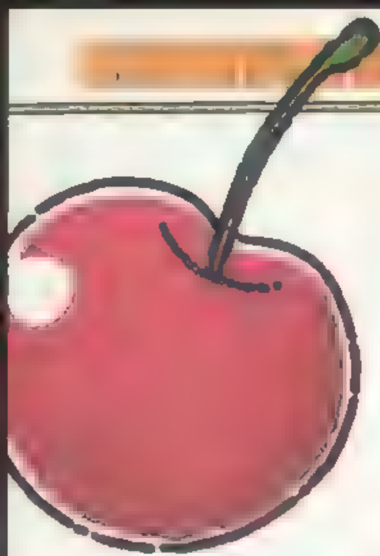
Depois de completar o jogo com esta segunda parte, puxe a manivela e tente obter três frutas iguais no caça-níqueis. Esse resultado pagará vinte vezes a sua aposta.

Segunda e última parte do nosso jogo caça-níqueis, este artigo nos coloca em condições de executar todo o programa, cujo primeiro segmento você já deve ter gravado.

S

APERTEM OS CINTOS

```
210 LET HOLD=0
220 LET TOTAL=TOTAL-10: LET NU
DGE=0: PRINT AT 13,26; INK 2;"
   "
230 IF HFLAG=0 THEN  LET HOLD=
0
240 FOR I=1 TO 3: FOR J=1 TO
12: SOUND .001,60
250 IF HOLD=0 THEN  PRINT AT 7
,10;A$(J);AT 7,15;B$(J);AT 7,
20;C$(J);AT 10,10;A$(J+1);AT
10,15;B$(J+1);AT 10,20;C$(J+1)
;AT 13,10;A$(J+2);AT 13,15;B$(
J+2);AT 13,20;C$(J+2): NEXT J:
 NEXT I
270 IF HOLD=1 THEN  PRINT AT 7
,15;B$(J);AT 7,20;C$(J);AT 10,
 15;B$(J+1);AT 10,20;C$(J+1);
AT 13,15;B$(J+2);AT 13,20;C$(J
+2): NEXT J: NEXT I
280 IF HOLD=6 THEN  PRINT AT 7
,15;B$(J);AT 10,15;B$(J+1);AT
13,15;B$(J+2): NEXT J: NEXT I
290 IF HOLD=2 THEN  PRINT AT 7
,10;A$(J);AT 7,20;C$(J);AT 10,
10;A$(J+1);AT 10,20;C$(J+1);AT
13,10;A$(J+2);AT 13,20;C$(J+2)
: NEXT J: NEXT I
300 IF HOLD=5 THEN  PRINT AT 7
,10;A$(J);AT 10,10;A$(J+1);AT
13,10;A$(J+2): NEXT J: NEXT I
310 IF HOLD=3 THEN  PRINT AT 7
,10;A$(J);AT 7,15;B$(J);AT 10,
10;A$(J+1);AT 10,15;B$(J+1);AT
 13,10;A$(J+2);AT 13,15;B$(J+2
): NEXT J: NEXT I
320 IF HOLD<>1 AND HOLD<>4 AND
HOLD<>6 THEN  LET M=INT (RND*
12): IF M=0 THEN  LET M=1
330 IF HOLD<>2 AND HOLD<>5 AND
```



```
HOLD<>4 THEN  LET K=INT (RND*
12): IF K=0 THEN  LET K=1
340 IF HOLD<>3 AND HOLD<>5 AND
HOLD<>6 THEN  LET L=INT (RND*
12): IF L=0 THEN  LET L=1
350 LET HOLD=0
360 PRINT AT 7,10;A$(M);AT 7,
15;B$(K);AT 7,20;C$(L);AT 10,
10;A$(M+1);AT 10,13;B$(K+1);AT
10,20;C$(L+1);AT 13,10;A$(M+2)
;AT 13,15;B$(K+2);AT 13,20;C$(
L+2)
```

A linha 210 coloca zero na variável que segura uma ou mais rodas, que podem ser escolhidas através do teclado. A rotina verifica qual é o conteúdo dessa variável e faz com que se movimentem as rodas que estão livres.

Depois de girar as rodas, a rotina recoloca zero em **HOLD** na linha 350 — os botões que seguram as rodas são apagados — e, em seguida, a linha 360 incumbe-se de desenhá-las em sua posição final, paradas.

VERIFIQUE O RESULTADO

```
370 GOSUB 510
510 LET T$=A$(M)+B$(K)+C$(L)
520 LET L$=A$(M+1)+B$(K+1)+C$(
L+1)
530 LET L$=A$(M+2)+B$(K+2)+C$(
L+2)
540 GOSUB 660
550 RETURN
680 LET TEMP-TOTAL
690 IF M$( TO 4)=M$(5 TO 8)
AND M$( TO 4)=M$(9 TO ) THEN
LET TOTAL-TOTAL+50: IF M$( TO
4)=C$(4) THEN  LET TOTAL=TOTAL
+5000: GOTO 640
700 IF M$( TO 4)=C$(1) AND M$(
TO 4)=M$(5 TO 8) AND M$( TO 4)
=M$(9 TO ) THEN  LET TOTAL=TOT
AL+50
710 IF M$( TO 4)=C$(3) AND M$(
TO 4)=M$(5 TO 8) AND M$( TO 4)
=M$(9 TO ) THEN  LET TOTAL=TOT
AL+50
720 IF M$( TO 4)=A$(3) THEN
LET TOTAL-TOTAL+10: IF M$(5 TO
 8)=A$(3) THEN  LET TOTAL=TOTA
L+10
```



```
730 IF TOTAL>TEMP THEN  FOR I=
1 TO 6: SOUND .05,50: NEXT I
740 LET DD=INT (TOTAL/100):
LET CC=TOTAL-(DD*100): PRINT
INK 2; PAPER 6;AT 17,0;"$
 ";AT 18,0;"C      "; PAPER 7;
BRIGHT 1;AT 17,1;" ";DD;AT 18,
1;" ";CC
750 IF TOTAL<1 THEN  GOTO 760
760 RETURN
```

A linha 370 salta para a sub-rotina da linha 510, que coloca as três linhas mostradas pelas rodas nas variáveis alfanuméricas **T$**, **M$** e **L$**. A linha do meio, **M$**, é a que vale para fins de contagem de pontos.

A sub-rotina que faz isso começa na linha 680. Ela verifica se ocorreu, em algum lugar, uma linha premiada, e soma o valor do prêmio ao patrimônio acumulado pelo jogador.

COMO DAR UM EMPURRÃOZINHO

```
380 IF M<7 OR K=L OR L>2 THEN
LET NUDGE=1: PRINT  BRIGHT 1;
PAPER 7; INK 2;AT 13,26;"NUDGE
"
390 LET HFLAG=INT (RND+.5): IF
HFLAG=1 THEN  FOR I=1 TO 19
STEP 5: PRINT AT 16,I; INK 6;
BRIGHT 1;"HOLD";: NEXT I
400 IF INKEY$<>"" THEN  GOTO
400
410 LET I$=INKEY$: IF I$=""
THEN  GOTO 410
420 IF I$=" " THEN  FOR I=9 TO
19 STEP 5: PRINT  INK 2;AT 16,
I;"    ": NEXT I: GOTO 210
430 IF I$="E" AND NUDGE=1 THEN
 GOSUB 600: LET NUDGE=0: PRINT
AT 13,26; INK 2;"     ": SOUND
 1,30: GOSUB 510: LET RN=INT (
RND*10): IF INT (RN/2)=RN/2
AND HFLAG<>1 THEN  LET NUDGE=1
: PRINT AT 13,26; INK 7;
BRIGHT 1;"NUDGE": GOTO 400
440 IF I$="Q" AND NUDGE=1 THEN
 GOSUB 560: LET NUDGE=0: PRINT
AT 13,26; INK 2;"     ": SOUND
.1,30: GOSUB 510: LET RN=INT (
RND*10): IF INT (RN/2)=RN/2
AND RN<3 THEN  LET NUDGE=1:
PRINT AT 13,26; INK 7; BRIGHT
1,"NUDGE": GOTO 400
450 IF I$="W" AND NUDGE=1 THEN
 GOSUB 580: LET NUDGE=0: PRINT
AT 13,26; INK 2;"     ": SOUND
.1,30: GOSUB 510
```

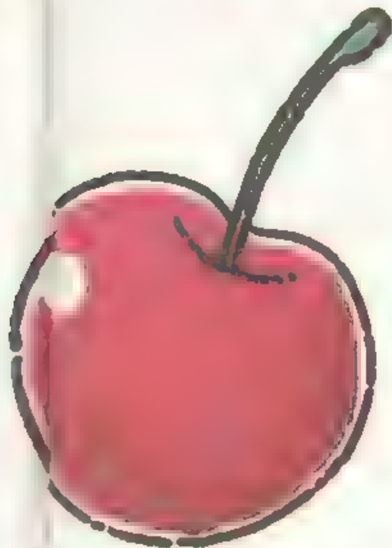


```
460 IF I$="D" AND NUDGE=1 THEN
 GOSUB 620: LET NUDGE=0: PRINT
AT 13,26; INK 2;"     ": SOUND
.1,30: GOSUB 510: LET RN=INT (
RND*10): IF INT (RN/2)<>RN/2
THEN  LET NUDGE=1: PRINT AT 13
,26; INK 7; BRIGHT 1;"NUDGE":
```









```
GOTO 400
470 IF I$="S" AND NUDGE=1 THEN
 GOSUB 660: LET NUDGE=0: PRINT
AT 13,26; INK 2;"      ": SOUND
 1,30: GOSUB 510: LET RN=INT (
RND*10): IF INT (RN/2)=RN/2
THEN  LET NUDGE=1: PRINT AT 13
,26; INK 7; BRIGHT 1;"NUDGE":
GOTO 400
480 IF I$="A" AND NUDGE=1 THEN
 GOSUB 640: LET NUDGE=0: PRINT
AT 13,26; INK 2;"      ": SOUND
.1,30: GOSUB 510: LET RN=INT (
RND*10): IF INT (RN/2)<>RN/2
AND RN>6 THEN  LET NUDGE=1:
PRINT AT 13,26; INK 7; BRIGHT
1;"NUDGE": GOTO 400
490 IF HFLAG=1 AND I$="1" OR I
$="2" OR I$="3" OR I$="4" OR I
$="5" OR I$="6" THEN  LET HOLD
=VAL I$: FOR I=1 TO 19 STEP 5:
 PRINT AT 16,I; INK 2;"     ":
 NEXT I: GOTO 220
500 GOTO 400
560 LET M=M+1: IF M>12 THEN
LET M=M-12
570 PRINT AT 7,10;A$(M);AT 10,
10;A$(M+1);AT 13,10;A$(M+2):
RETURN
580 LET K=K+1: IF K>12 THEN
LET K=K-12
590 PRINT AT 7,15;B$(K);AT 10,
15;B$(K+1);AT 13,15;B$(K+2):
RETURN
600 LET L=L+1: IF L>12 THEN
LET L=L-12
610 PRINT AT 7,20;C$(L);AT 10,
20;C$(L+1);AT 13,20;C$(L+2):
RETURN
620 LET L=L-1: IF L<1 THEN
LET L=L+12
630 PRINT AT 7,20;C$(L);AT 10,
20;C$(L+1);AT 13,20;C$(L+2):
RETURN
640 LET M=M-1: IF M<1 THEN
LET M=M+12
650 PRINT AT 7,10;A$(M);AT 10,
10;A$(M+1);AT 13,10;A$(M+2):
RETURN
660 LET K=K-1: IF K<1 THEN
LET K=K+12
670 PRINT AT 7,15;B$(K);AT 10,
15;B$(K+1);AT 13,15;B$(K+2):
RETURN
```



A rotina que empurra as rodas é muito semelhante àquela que as segura. As rodas são movimentadas para cima ou para baixo conforme a escolha do jogador. As teclas que empurram as rodas são mostradas na tela. A cada empurrão, um número aleatório é usado para determinar se o jogador poderá recorrer novamente a essa função.

RAPA-TUDO

```
760 CLS : PRINT AT 10,0;"
       FIM DE JOGO             VO
CE PERDEU TODO SEU DINHEIRO":
SOUND 1,-20
800 PRINT '"      QUER RECOMECA
R  (S/N) ?"
810 IF INKEY$="" THEN  GOTO
 810
820 LET I$=INKEY$: IF I$="S"
OR I$="s" THEN  RUN
830 STOP
840 CLS : PRINT AT 10,0;"
      PARABENS !               VO
CE ACABA DE GANHAR O PREMIO.":
 PRINT '" VOCE ESTA $500,00 MA
IS RICO !": FOR J=1 TO 3: FOR
I=1 TO 10: SOUND .01,5*I: NEXT
 I: NEXT J
850 GOTO 800
```

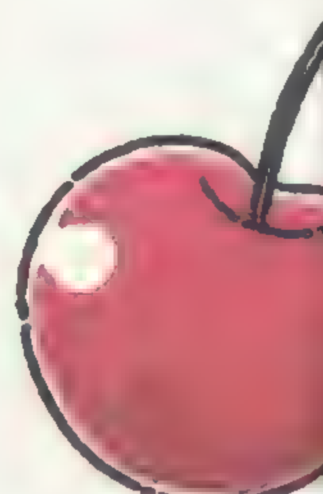
A rotina da linha 760 oferece ao usuário a oportunidade de disputar mais um jogo. Ela é chamada quando o jogador fica sem dinheiro.

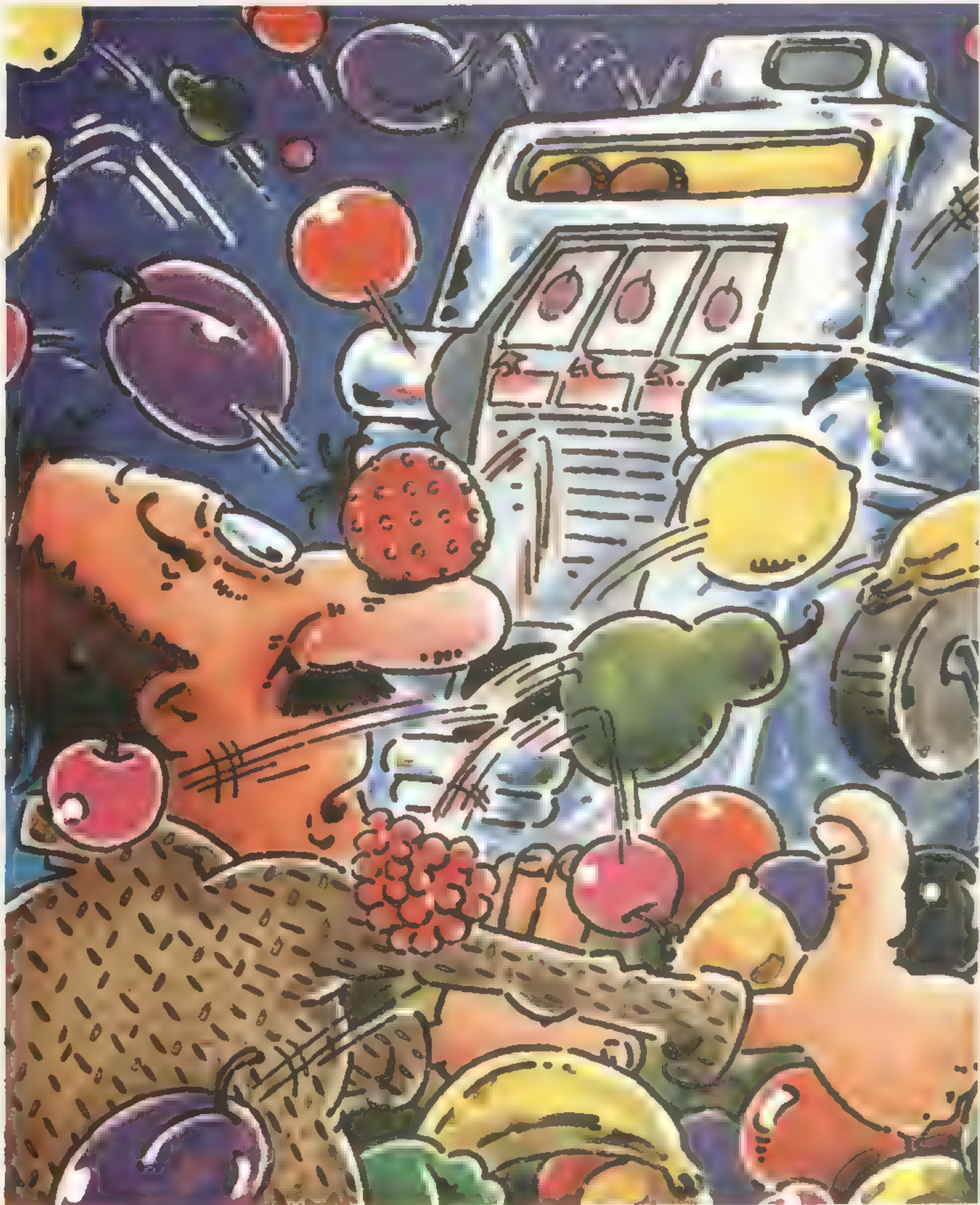
A rotina que cuida do prêmio máximo — três sinos — dá as boas novas ao jogador e soma $500 ao seu score. Nesse caso, a partida termina, pois a banca foi quebrada. O jogador pode, então, jogar outra vez.



A ROTINA PRINCIPAL

```
350 M=100:H=-1:I=-1:J=-1:P=RND(
16)-1:Q=RND(16)-1:R=RND(16)-1
360 SCREEN 1:GOSUB 1000:GOSUB 2
000:GOSUB 2500:IF M>0 THEN 360
370 CLS:PRINT @101,"YOU RAN OUT
 OF MONEY"
380 PRINT @417," <SPACE> PARA R
ECOMECAR"
390 IF INKEY$<>" " THEN 390 ELS
E RUN
```





A linha 350 estabelece o valor do primeiro cacife em dólar. Ela também acerta os valores iniciais das variáveis que indicam se alguma roda está presa, assim como das variáveis que controlam a posição das rodas. A linha 360, laço principal do programa, chama ordenadamente as sub-rotinas que giram as rodas, desenham as frutas na tela e permitem ao jogador apostar, segurar e empurrar as rodas.

Se o jogador ficar sem dinheiro, a linha 370 terminará o jogo e oferecerá ao usuário a oportunidade de disputar outra partida.

FRUTAS NA TELA

```
500 ON CH+1 GOTO 540,530,560,55
0,570,510,520
510 PUT(XX,YY)-(XX+31,YY+15),B,
PSET:RETURN
520 PUT(XX,YY)-(XX+31,YY+15),C,
PSET:RETURN
530 PUT(XX,YY)-(XX+31,YY+15),A,
PSET·RETURN
540 PUT(XX,YY)-(XX+31,YY+15),BR
,PSET:RETURN
550 PUT(XX,YY)-(XX+31,YY+15),S,
PSET·RETURN
560 PUT(XX,YY)-(XX+31,YY+15),PL
,PSET:RETURN
570 PUT(XX,YY)-(XX+31,YY+15),P,
PSET:RETURN
1000 M=M-10.FOR L=1 TO RND(3)+R
ND(3)
1010 IF H GOSUB 1520
1020 IF I GOSUB 1530
1030 IF J GOSUB 1540
1040 NEXT:IF H THEN SOUND 100,1
1050 FOR L=1 TO RND(3)+RND(3)
1060 IF I GOSUB 1530
1070 IF J GOSUB 1540
1080 NEXT:IF I THEN SOUND 120,1
1090 FOR L=1 TO RND(3)+RND(3)
1100 IF J GOSUB 1540
1110 NEXT IF J THEN SOUND 140,1
1120 H=-1:I=-1·J=-1:RETURN
1500 CLS:IF D=0 THEN RETURN ELS
E PRINT @166,USING"CREDITO= $$#
## ##";M/100:FOR A=10 TO D STEP
 10·M=M+10·PRINT @166,USING"CRE
DITO= $$###.##";M/100
1510 SOUND 200,1:FOR B=0 TO 400
:NEXT B,A:RETURN
1520 P=(P-1) AND 15·XX=48·YY 28
:FOR G=P-1 TO P+1:CH=R1(15 AND
G):GOSUB 500:YY=YY+32:NEXT·RETU
RN
1530 Q=(Q-1) AND 15:XX=112·YY=2
8:FOR G=Q-1 TO Q+1:CH=R2(15 AND
 G):GOSUB 500:YY=YY+32:NEXT:RET
URN
1540 R=(R-1) AND 15·XX=176:YY=2
8:FOR G=R-1 TO R+1:CH=R3(15 AND
 G):GOSUB 500:YY=YY+32:NEXT:RET
URN
1550 C=9·IF (R1(P)=R2(Q)) AND(R
2(Q)=R3(R)) THEN C=R1(P):RETURN
1560 IF R1(P)=R2(Q) AND (R1(P)=
6 OR R1(P)=5) THEN C=1+R1(P):RE
TURN
1570 IF R1(P)=6 THEN C=8
1580 RETURN
```

As linhas 500 a 570 usam PUT para desenhar as frutas na tela. As linhas 1000 a 1120 giram as rodas, chamando as sub-rotinas das linhas 1520 a 1540. As linhas 1150 a 1180 acusam um resultado premiado e a linha 1500 soma o prêmio às posses do jogador.

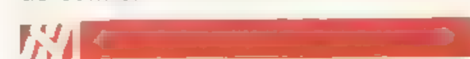
APOSTAR, SEGURAR, EMPURRAR

```
2000 GOSUB 1550
2010 IF C=9 OR C=0 THEN D=W(C):
GOSUB 1500:RETURN
2020 CLS9-C:PRINT @265,"apostar
";:PRINT@278,USING "$$#.## ";W(
C)/100:
2030 PLAY "L4T20B":PRINT @212,U
SING"$$#.## ";W(C-1)/100;.PRINT
 @340,STRING$(7,271-C*16);
2040 PLAY"T20C":PRINT @212,STRI
NG$(7,271-C*16);:PRINT @340,USI
NG"$$#.## ";W(C+1)/100;
2050 R$=INKEY$ IF R$<>" " AND R
$<>CHR$(13) THEN 2030
2060 IF R$=CHR$(13) THEN CLS:D=
W(C):GOSUB 1500:RETURN
2070 IF RND(2)=1 THEN CLS:D=W(C
+1)·GOSUB 1500:RETURN
2080 C=C-1:IF C=0 THEN D=200:GO
SUB 1500:RETURN
2090 GOTO 2020
2500 IF RND(4)=1 GOSUB 3060:GOT
O 2550
2510 IF RND(5)<3 THEN 2560
2520 FOR K=1 TO 2000:NEXT:SCREE
N 1,0
2530 A$=INKEY$:IF A$<>" " AND A
$<>"C" THEN 2530
2540 IF A$=" " THEN RETURN
2550 CLS:PRINT @166,USING"CREDI
TO= $$### ##";M/100:GOTO 2520
2560 SCREEN 1,0·H=-1:I=-1·J=-1
2570 IF H THEN PUT(38,122)-(91,
143),H,NOT
2580 IF I THEN PUT(102,122)-(15
5,143),H,NOT
2590 IF J THEN PUT(166,122)·(21
9,143),H,NOT
2600 R$=INKEY$:IF R$=" " THEN F
OR K=0 TO 2·PUT(38+64*K,122)-(9
1+64*K,143),H,PSET NEXT·RETURN
3000 IF R$<"1" OR R$>"4" THEN 2
570
3010 ON VAL(R$) GOTO 3020,3030,
3040,3050
3020 H=-1:I=-1·J=-1:GOTO 2570
3030 H=0:PUT(38,122)-(91,143),H
,PSET:GOTO 2570
3040 I=0:PUT(102,122)-(155,143)
,H,PSET:GOTO 2570
3050 J=0:PUT(166,122)-(219,143)
,H,PSET:GOTO 2570
3060 SCREEN 1,0:COLOR 4,2:PUT(1
59,156)-(224,170),H,NOT:PLAY"L4
0T10"
3070 K=1
3080 LINE(10+K*16,158)-(21+K*16
,169),PRESET,BF
3090 IF INKEY$=" " THEN 3120
3100 K=K+1:PLAY STR$(K*2):IF K<
6 THEN 3080
3110 FOR K=1 TO 5:LINE(10+K*16,
158)-(21+K*16,169),PSET,BF:NEXT
:GOTO 3070
3120 N=K:PUT(159,156)-(224,170)
,H,NOT
3130 R$=INKEY$:IF(R$<"5" OR R$>
"9") AND R$<>"0" THEN 3130
3140 IF R$="0" THEN R$="10"
3150 ON VAL(R$)-4 GOTO 3160,317
0,3180,3190,3200,3210
3160 P=P+2:GOSUB 1520:GOTO 3220
3170 Q=Q+2:GOSUB 1530.GOTO 3220
3180 R=R+2:GOSUB 1540:GOTO 3220
3190 GOSUB 1520:GOTO 3220
3200 GOSUB 1530:GOTO 3220
3210 GOSUB 1540
3220 SOUND 40,1:GOSUB 1550:IF C
<9 GOSUB 2010:N=0 GOTO 3250
3230 IF N=1 THEN N=0:GOTO 3250
3240 LINE(10+N*16,158)-(21+N*16
,169),PSET,BF:N=N-1:GOTO 3130
3250 FOR K=1 TO 5:LINE(10+K*16,
158)-(21+K*16,169),PSET,BF:NEXT
·RETURN
```

As linhas 2010 a 2050 permitem que o jogador faça a sua aposta. Ele pode então melhorar o prêmio que recebeu anteriormente (ou, na pior das hipóteses, perder parte dele).

A rotina que segura as rodas encontra-se entre as linhas 2530 e 3050, sendo chamada da linha 2510. Os indicadores que definem a liberdade das rodas são modificados de acordo com as teclas apertadas pelo jogador.

As linhas 3060 a 3250 permitem que o jogador empurre uma das rodas, movendo-a uma posição para cima ou para baixo. O número de movimentos oferecidos é selecionado ao acaso na linha 2500. A rotina detecta as teclas selecionadas, movendo as rodas de acordo com cada uma delas.



A ROTINA PRINCIPAL

```
1160 FOR I=1 TO 2000·NEXT
1200 R=RND(-TIME)·M=100 H= 1·I=
-1:J=-1:P=INT(RND(1)*15):Q=INT(
RND(1)*15):R=INT(RND(1)*15)
1210 LOCATE 0,22·PRINT STRING$(
30,32),·GOSUB 1400 GOSUB 2000·G
OSUB 2500·IF M>0 AND M<5000 THE
N 1210
1215 IF M>=5000 THEN CLS·LOCATE
 0,11·PRINT "RAPA TUDO E ESTOUR
A A BANCA":GOTO 1230
1220 LOCATE 0,22·PRINT "SEU DIN
HEIRO ACABOU           ";
1230 LOCATE 0,23·PRINT "APERTE
>ESPACO P/ REPETIR";
1240 IF INKEY$<>" " THEN 1240 E
LSE RUN
```

A linha 1200 estabelece o valor de um dólar para o primeiro cacife e acerta os valores iniciais das variáveis que indicam se alguma das rodas está presa, assim como das variáveis que controlam a posição das rodas. A linha 1210 é o laço principal do programa. Ela chama ordenadamente as sub-rotinas que giram as rodas, desenham as frutas na tela e permitem ao jogador apostar, segurar e empurrar as rodas. O gerador de números randômicos também é acertado nessa linha.

Se o jogador ficar sem dinheiro, a linha 1220 concluirá o jogo e dará ao usuário a oportunidade de jogar novamente. Quando existirem três sinos na linha correspondente ao meio das rodas, haverá o estorno da banca e o jogo também terá chegado ao fim, agora com a vitória do jogador.

FRUTAS NA TELA

```
1300 VPOKE BASE(5)+YY,C1:VPOKE
BASE(5)+YY+1,C2:RETURN
1400 M=M-10:GOSUB 4000:FOR L=1
TO INT(RND(1)*3+1)*INT(RND(1)*3
+1)
1405 IF H THEN GOSUB 1520
1410 IF I THEN GOSUB 1530
1415 IF J THEN GOSUB 1540
1420 NEXT
1425 FOR L=1 TO INT(RND(1)*3+1)
*INT(RND(1)*3+1)
1430 IF I THEN GOSUB 1530
1435 IF J THEN GOSUB 1540
1440 NEXT
1450 FOR L=1 TO INT(RND(1)*3+1)
*INT(RND(1)*3+1)
1455 IF J THEN GOSUB 1540
1460 NEXT
1470 FOR L=8 TO 10:SOUND L,0:NE
XT:H=-1:I=-1:J=-1
1480 RETURN
1500 IF D=0 THEN RETURN ELSE M=
M+D
1510 RETURN
1520 P=(P-1)AND15:YY=235:FOR G=
P-1 TO P+1:C1=A(15ANDG,1):C2=A(
15ANDG,2):GOSUB 1300:YY=YY+96:N
EXT:RETURN
1530 Q=(Q-1)AND15:YY=239:FOR G=
Q-1 TO Q+1:C1=B(15ANDG,1):C2=B(
15ANDG,2):GOSUB 1300:YY=YY+96:N
EXT:RETURN
1540 R=(R-1)AND15:YY=243:FOR G=
R-1 TO R+1:C1=C(15ANDG,1):C2=C(
15ANDG,2):GOSUB 1300:YY=YY+96:N
EXT:RETURN
1550 D=0
1551 IF A(P,1)=136 THEN D=10
1552 IF A(P,1)=B(Q,1) AND A(P,1
)=136 THEN D=20:RETURN
1553 IF NOT(A(P,1)=B(Q,1) AND B
(Q,1)=C(R,1)) THEN RETURN
1554 A=A(P,1)
1556 IF A=120 THEN D=50
1557 IF A=184 THEN D=50
1558 IF A=216 THEN D=50
1559 IF A=152 THEN D=100
1560 IF A=168 THEN D=100
1561 IF A=200 THEN M=M+5000:GOT
O 1215
1570 RETURN
```

A linha 1300 usa o comando **VPOKE** para desenhar as frutas na tela. As linhas que vão de 1400 a 1480 giram as rodas chamando as sub-rotinas das linhas 1520 a 1540.

As linhas 1550 a 1570, por sua vez, verificam se houve um resultado premiado e a linha 1500, finalmente, soma o prêmio às posses do jogador.

EMPURRAR E SEGURAR AS RODAS

```
2000 GOSUB 1550:GOSUB 1500:GOSU
B 2550:RETURN
2500 IF INT(RND(1)*4+1)=1 THEN
GOSUB 3060:GOSUB 2550:GOTO 2530
2510 IF INT(RND(1)*5+1)<3 THEN
2560
2530 A$=INKEY$:IF A$<>" " THEN
2530
2540 RETURN
2550 M$=STR$(M):LOCATE 2,15:PRI
NT RIGHT$(M$,2);:LOCATE 2,14:PR
INT "  ";:IF M>=100THEN LOCATE
2,14:PRINT LEFT$(M$,2);:RETURN
2555 RETURN
2560 H=-1:I=-1:J=-1:LOCATE 4,22
:PRINT "PODE SEGURAR AS RODAS"
2600 R$=INKEY$:IF (R$<"1" OR R$
>"6") AND R$<>" " THEN 2560
3000 IF R$=" " THEN RETURN
3010 ON VAL(R$) GOTO 3020,3025,
3030,3035,3040,3045
3020 H=0:GOTO 3050
3025 I=0:GOTO 3050
3030 J=0:GOTO 3050
3035 H=0:I=0:GOTO 3050
3040 J=0:I=0:GOTO 3050
3045 J=0:H=0
3050 RETURN
3060 LOCATE 3,22:PRINT "PODE EM
PURRAR AS RODAS"
3070 N=INT(RND(1)*5+1)
3080 R$=INKEY$:IF R$="" THEN 30
80
3100 IF R$="Q" THEN 3160
3110 IF R$="W" THEN 3170
3120 IF R$="E" THEN 3180
3130 IF R$="A" THEN 3190
3140 IF R$="S" THEN 3200
3150 IF R$="D" THEN 3210
3155 GOTO 3080
3160 P=P+2:GOSUB 1520:GOTO 3220
3170 Q=Q+2:GOSUB 1530:GOTO 3220
3180 R=R+2:GOSUB 1540:GOTO 3220
3190 GOSUB 1520:GOTO 3220
3200 GOSUB 1530:GOTO 3220
3210 GOSUB 1540
3220 GOSUB 1550:IF D>0 THEN GOS
UB 2000:N=0:GOTO 3250
3230 IF N=1 THEN N=0:GOTO 3250
3240 N=N-1:GOTO 3080
3250 LOCATE 0,22:PRINT STRING$(
30,32);:RETURN
```

A rotina que segura as rodas está entre as linhas 2560 e 3050, sendo chamada a partir da linha 2510. Os indicadores que revelam se as rodas estão livres são modificados conforme as teclas apertadas pelo jogador.

As linhas que vão de 3060 a 3250 permitem que o jogador empurre uma das rodas, movendo-a uma posição para cima ou para baixo. O número de empurrões oferecidos é selecionado ao acaso na linha 3070. A rotina detecta as teclas selecionadas, movendo as rodas de acordo com cada uma delas.

EFEITOS SONOROS

A rotina situada na linha 4000 acerta os valores dos registros do chip sonoro do MSX para simular o ruído de rodas em movimento.

```
4000 RESTORE 4030:FOR L=4 TO 13
4010 READ A:SOUND L,A
4020 NEXT
4030 DATA 0,9,15
4040 DATA 42,0,12,16,100,5,12
4050 RETURN
```

```
300  GOSUB 700: GOSUB 4000: GOS
UB 310: GOSUB 780: GOTO 1000
1000 M = 1:H = 1:I = 1:J = 1:P
=  INT ( RND (1) * 16):Q =  INT
 ( RND (1) * 16):R =  INT ( RND
 (1) * 16)
1010  POKE  - 16299,0: POKE  -
16304,0: GOSUB 1300: GOSUB 2000
: GOSUB 2500: IF M > 0 THEN 101
0
1020  TEXT : HOME : VTAB 20: PR
INT "SEU DINHEIRO ACABOU"
1030  PRINT "APERTE A BARRA DE
ESPACO": PRINT "PARA JOGAR NOVA
MENTE"
1040  GET A$: IF A$ = " " THEN
 RUN
1050  GOTO 1040
1200  HCOLOR= 0: DRAW 15 AT XX,
YY: DRAW 15 AT XX + 8,YY: DRAW
15 AT XX,YY + 4: DRAW 15 AT XX
+ 8,YY + 4
1205  ON CH + 1 GOTO 1240,1230,
1260,1250,1270,1210,1220
1210  HCOLOR= 3: DRAW 7 AT XX,Y
Y: DRAW 8 AT XX + 7,YY: RETURN
1220  HCOLOR= 3: DRAW 5 AT XX,Y
Y: DRAW 6 AT XX + 7,YY: RETURN
1230  HCOLOR= 7: DRAW 13 AT XX,
YY: DRAW 14 AT XX + 7,YY: RETUR
N
1240  HCOLOR= 3: DRAW 3 AT XX,Y
Y: DRAW 4 AT XX + 8,YY: RETURN
1250  HCOLOR= 3: DRAW 1 AT XX +
 1,YY: DRAW 2 AT XX + 8,YY: RET
URN
1260  HCOLOR= 7: DRAW 11 AT XX,
YY: DRAW 12 AT XX + 7,YY: RETUR
N
```

```
1270  HCOLOR= 3: DRAW 9 AT XX +
 1,YY: DRAW 10 AT XX + 8,YY: RE
TURN
1300 M = M - .1: FOR L = 1 TO
INT (3 *  RND (1) + 1) +  INT (
3 *  RND (1) + 1)
1310  IF H THEN  GOSUB 1520:X =
  PEEK ( - 16336)
1320  IF I THEN  GOSUB 1530:X =
  PEEK ( - 16336)
1330  IF J THEN  GOSUB 1540:X =
  PEEK ( - 16336)
1340  NEXT : IF H THEN  CALL  -
 198
1350  FOR L = 1 TO  INT (3 *  R
ND (1) + 1) +  INT (3 *  RND (1
) + 1)
1360  IF I THEN  GOSUB 1530:X =
  PEEK ( - 16336)
1370  IF J THEN  GOSUB 1540:X =
  PEEK ( - 16336)
1380  NEXT : IF I THEN  CALL  -
 198
1390  FOR L = 1 TO  INT (3 *  R
ND (1) + 1) +  INT (3 *  RND (1
) + 1)
1400  IF J THEN  GOSUB 1540:X =
  PEEK ( - 16336)
1410  NEXT : IF J THEN  CALL  -
 198
1420 H = 1:I = 1:J = 1: RETURN
1500  IF D = 0 THEN  RETURN
1505  HOME : TEXT : VTAB 20:M =
 M + D: PRINT "CREDITO: $";M
1510  FOR B = 0 TO 400: NEXT :
RETURN
1520 P = P - 1:XX = 80:YY = 90:
 IF P =  - 1 THEN P = 15
1522  FOR G = P - 1 TO P + 1: I
F G =  - 1 THEN CH = R1(15): GO
TO 1525
1523  IF G = 16 THEN CH = R1(0)
: GOTO 1525
1524 CH = R1(G)
1525  GOSUB 1200:YY = YY + 20:
NEXT : RETURN
1530 Q = Q - 1:XX = 128:YY = 90
: IF Q =  - 1 THEN Q = 15
1532  FOR G = Q - 1 TO Q + 1: I
F G =  - 1 THEN CH = R2(15): GO
TO 1535
1533  IF G = 16 THEN CH = R2(0)
: GOTO 1535
1534 CH = R2(G)
1535  GOSUB 1200:YY = YY + 20:
NEXT : RETURN
1540 R = R - 1:XX = 174:YY = 90
: IF R =  - 1 THEN R = 15
1542  FOR G = R - 1 TO R + 1: I
F G =  - 1 THEN CH = R3(15): GO
TO 1545
1543  IF G = 16 THEN CH = R3(0)
: GOTO 1545
1544 CH = R3(G)
1545  GOSUB 1200:YY = YY + 20:
NEXT : RETURN
1550 C = 9: IF (R1(P) = R2(Q))
AND (R2(Q) = R3(R)) THEN C = R1
(P): RETURN
1560  IF R1(P) = R2(Q) AND (R1(
P) = 6 OR R1(P) = 5) THEN C = 1
 + R1(P): RETURN
1570  IF R1(P) = 6 THEN C = 8
1580  RETURN
2000  GOSUB 1550
2010  IF C = 9 OR C = 0 THEN D
= W(C): GOSUB 1500: RETURN
2020  HOME : TEXT : VTAB 20: PR
INT "PREMIO = $";W(C)
2030  PRINT "QUER APOSTAR ?
2050  PRINT "<ESPACO>": TAB( 20
);"APOSTA": PRINT "<RETURN>"; T
AB( 20);"RECOLHE O PREMIO"
2055  GET A$: IF A$ <  > " " AN
D A$ <  >  CHR$ (13) THEN 2020
2060  IF A$ =  CHR$ (13) THEN
HOME :D = W(C): GOSUB 1500: RET
URN
2070  IF  INT (2 *  RND (1) + 1
) = 2 THEN  HOME :D = W(C + 1):
 GOSUB 1500: RETURN
2080 C = C - 1: IF C = 0 THEN D
 = 2: GOSUB 1500: RETURN
2090  GOTO 2020
2500  IF  INT (4 *  RND (1) + 1
) = 1 THEN  GOSUB 3060: GOTO 25
50
2510  IF  INT (5 *  RND (1) + 1
) < 3 THEN 2560
2530  GET A$: IF A$ <  > "C" TH
EN  RETURN
2550  HOME : TEXT : VTAB 20: PR
INT "CREDITO  = $";M: GOTO 2530
2560 H = 1:I = 1:J = 1
2565  HOME : TEXT
2570  IF  NOT H THEN  VTAB 20:
HTAB 10: PRINT "HOLD";
2580  IF  NOT I THEN  VTAB 20:
HTAB 20: PRINT "HOLD";
2590  IF  NOT J THEN  VTAB 20:
HTAB 30: PRINT "HOLD";
2595  VTAB 22: HTAB 1: PRINT "V
OCE PODE PRENDER AS RODAS";
2597  FOR K = 1 TO 4000: NEXT :
 POKE  - 16299,0: POKE  - 16304
,0: FOR K = 1 TO 4000: NEXT : P
OKE  - 16300,0: POKE  - 16303,0
2600  GET A$: IF A$ = " " THEN
 VTAB 20: PRINT " ": RETURN
3000  IF A$ < "1" OR A$ > "4" T
HEN 2570
3010  ON  VAL (A$) GOTO 3020,30
30,3040,3050
3020 H = 1:I = 1:J = 1:GOTO 2565
3030 H = 0: GOTO 2565
3040 I = 0: GOTO 2565
3050 J = 0: GOTO 2565
3060  HOME : TEXT : VTAB 22: PR
INT "PODE EMPURRAR AS RODAS";:
GET A$
3070 N =  INT (5 *  RND (1) + 1
)
3080  POKE  - 16299,0: POKE  -
16304,0: GET A$: IF (A$ < "5" O
R A$ > "9") AND A$ <  > "0" THE
N 3060
3140  IF A$ = "0" THEN A$ = "10
"
3150  ON  VAL (A$) - 4 GOTO 316
0,3170,3180,3190,3200,3210
3160 P = P + 2: GOSUB 1520:X =
 PEEK ( - 16336): GOTO 3220
3170 Q = Q + 2: GOSUB 1530:X =
 PEEK ( - 16336): GOTO 3220
3180 R = R + 2: GOSUB 1540:X =
 PEEK ( - 16336): GOTO 3220
3190  GOSUB 1520:X =  PEEK ( -
16336): GOTO 3220
3200  GOSUB 1530:X =  PEEK ( -
16336): GOTO 3220
3210  GOSUB 1540
3220  GOSUB 1550: IF C < 9 THEN
 GOSUB 2010:N = 0: GOTO 3250
3230  IF N = 1 THEN N = 0: GOTO
 3080
3240 N = N - 1: GOTO 3080
3250  RETURN
```

A linha 1000 estabelece que o valor do cacife inicial é um dolar. Ela tambem acerta os valores iniciais das variáveis que indicam se alguma roda está presa, assim como das variáveis que controlam a posição das rodas. A linha 1010, laço principal do programa, chama ordenadamente as sub-rotinas que giram as rodas, desenham as frutas na tela e permitem ao jogador apostar, segurar e empurrar as rodas.

Se o jogador ficar sem dinheiro, a linha 1020 terminará o jogo, permitindo-lhe jogar novamente.

As linhas 1200 a 1270 usam **DRAW** para desenhar as frutas na tela. As linhas 1300 a 1420 giram as rodas, chamando as sub-rotinas das linhas 1520 a 1540. As linhas 1550 a 1580 acusam um resultado premiado e a 1500 soma o prêmio às posses do jogador.

As linhas 2010 a 2050 permitem que o jogador aposte. Este pode então melhorar o prêmio que recebeu (ou perder parte dele). A rotina que segura as rodas se encontra entre as linhas 2530 e 3050, sendo chamada a partir da linha 2510. Os indicadores que revelam se as rodas podem girar livremente são modificados conforme as teclas apertadas pelo jogador.

As linhas 3060 a 3250 possibilitam ao jogador empurrar uma das rodas, movendo-a uma posição para cima ou para baixo. O número de empurrões oferecidos é selecionado ao acaso na linha 2500. A rotina detecta as teclas selecionadas, movendo as rodas de acordo com cada uma destas.



Para funcionar no TK-2000, o programa precisa de algumas modificações.

Elimine todos os **POKE −16299,0** e **POKE −16304,0**. Substitua os comandos **HOME:TEXT** — sempre que aparecerem assim juntos — por **GOSUB 5000**.

Acrescente ainda as linhas:

```
5000  FOR K = 160 TO 191
5010  HCOLOR= 0
5020  HPLOT 0,K TO 279,K
5030  NEXT : RETURN
```

MENSAGENS SECRETAS

6 de junho de 1944: os acordes iniciais da Quinta Sinfonia de Beethoven dão o sinal para a invasão da Normandia pelos aliados, num dos casos mais famosos de mensagem em código.

Embora sem saber, a maioria das pessoas usa códigos para comunicar-se em seu dia-a-dia. Assim, quando alguém pede um sapato número 39 ou fala de um microprocessador Z80, está empregando um código. Do mesmo modo, um cientista nuclear usa equações muito complicadas, que, na verdade, não passam de códigos, para simplificar a representação de fenômenos complexos. Nós poderíamos obter os mesmos resultados simplesmente descrevendo tais fenômenos por meio de palavras. No entanto, isso nos tomaria muito mais tempo.

Nos exemplos citados, a preocupação maior consiste em facilitar a comunicação e não em esconder a informação transmitida. Os códigos podem também ser usados para economizar gastos ou espaços de armazenamento. Imaginemos, por exemplo, uma companhia que realiza muitos contratos envolvendo cláusulas e parágrafos iguais. Se todos esses detalhes fossem guardados em fitas ou discos, economizaríamos certamente muito espaço, codificando as partes que se repetem várias vezes.

A ciência que estuda as mensagens secretas chama-se criptografia. Seu nome é formado a partir de duas palavras gregas — *Kryptos* (secreto) e *graphos* (escrita) —, e refere-se tanto à escrita codificada quanto à cifrada. Os termos código e cifra têm, na realidade, significado ligeiramente diferente no que diz respeito ao modo como a mensagem é transmitida. Quando a informação é comunicada letra por letra, dizemos que ela está cifrada. Quando uma ou mais palavras são transformadas em outro grupo de expressões ou de números, por intermédio de alguma espécie de dicionário, dizemos que ela está codificada. Na prática, o termo codificar é usado em ambos os casos.

CÓDIGOS SECRETOS

A codificação e decodificação de mensagens secretas já foram de uso restrito dos militares, ou pelo menos dos Serviços de Inteligência. Hoje, porém, o uso de linhas públicas de telefone e de satélites para transmissão de informações confidenciais fez aumentar enormemente a necessidade dos códigos.

Uma das primeiras aplicações importantes dos computadores aconteceu durante a Segunda Guerra Mundial, quando equipamentos IBM foram utilizados pelos Aliados para decifrar mensagens em código do inimigo. Desde então, espiões e contra-espiões das grandes potências têm acompanhado atentamente os avanços tecnológicos nesse campo.

Atualmente, as possibilidades abertas pela computação são tantas que jovens pouco escrupulosos — denominados *hackers* — ligam seus micros a redes telefônicas e penetram facilmente em sistemas bancários e instalações militares, quebrando códigos e alterando informações.

Neste e no próximo artigo mostraremos como empregar o computador para criar mensagens em código secreto, recorrendo a diversos métodos que, como cada tipo de espionagem, têm diferentes níveis de sofisticação.

Mesmo não sendo um agente internacional, você pode lançar mão desses métodos quando quiser enviar mensagens confidenciais a amigos que também tenham micros. E aqui vai um desafio. Neste artigo está escondida uma informação codificada. Tente encontrá-la.

CÓDIGOS DE POSIÇÃO

Os símbolos mostrados nas telas de vídeo da página 889 parecem não ter significado algum, mas na realidade são exemplos de códigos de posição. Como o nome sugere, esse é um código baseado na posição de um símbolo em relação a um ponto dado.

Esse tipo de código foi usado pela primeira vez há mais de 2000 anos por Lisandro, um almirante espartano que, durante uma das batalhas da Guerra do Peloponeso (travada entre as cidades gregas Esparta e Atenas), percebeu que, no cinto de um de seus escravos, a distância dos furos até a fivela continha na realidade uma mensagem secreta. Desde então, os códigos tornaram-se um dos recursos mais poderosos da chamada "arte de guerra".

Você pode usar um tipo de código de posição simplesmente escrevendo o alfabeto no alto de uma folha e colocando abaixo dele as distâncias recebidas, conforme mostram as figuras 1 e 2. Como você tem a letra-chave na primeira linha, é fácil decifrar a mensagem. Para dificultar sua decodificação, basta fazer uma rotação nas letras-chave (veja a figura 3). Você pode, por exemplo, começar com N e, quando chegar no Z, começar de novo com A. Isto é chamado rotação cíclica.

O primeiro programa usa esse método para produzir uma versão codificada de seu texto — entretanto, não devem ser deixados espaços em branco entre as palavras. Se você quiser realmente enviar sua mensagem para alguém, vai precisar de uma impressora.



S

```
20 BORDER 0: PAPER 0: INK 7:
CLS
30 PRINT TAB 6;" Codigo de po
sicao ": PRINT
40 PRINT  INK 2; PAPER 7;
FLASH 1;AT 6,10;"  CUIDADO  "
: PRINT
50 PRINT "Nao deixe espacos e
ntre palavras"
60 PRINT . PRINT "       Escre
va em minusculas"
70 INPUT "Qual a mensagem ?"'
a$
80 FOR i=1 TO 400· NEXT i:
CLS
90 FOR i=1 TO LEN (a$)
100 LET b$=a$(i)
110 LET v=CODE (b$)-96
120 IF v<=32 THEN  PRINT TAB v
; INK 6;"*": GOTO 150
130 LET v=v-26
140 PRINT TAB (v); INK 6;"*"
150 NEXT i
```



```
20 CLS
30 PRINT @6,"CODIGO DE DISTANCI
A"
40 PRINT @140,"CUIDADO":PRINT
50 PRINT"NAO DEIXE ESPACOS ENTR
E PALAVRAS"
60 PRINT·PRINT
70 PRINT"QUAL A MENSAGEM ?".INP
UT A$
80 FOR I=1 TO 600·NEXT:CLS
90 FOR I=1 TO LEN(A$)
```





1 Código de posição com distância horizontal



2 Código de posição com distância vertical

```
100 B$=MID$(A$,I,1)
110 V=ASC(B$)-45
120 IF V<=32 THEN PRINT TAB(V)
"*"·GOTO 150
130 V=V-26
140 PRINT TAB(V)"*"
150 NEXT
```



```
20 CLS
30 PRINT TAB(10) "código de pos
ição":PRINT
40 PRINT TAB(14) "cuidado"·PRIN
T
50 PRINT TAB(4) "não deixe espa
ço entre as palavras"
60 PRINT·PRINT
70 PRINT"qual a sua mensagem?"·
INPUT A$
80 FOR I=1 TO 600·NEXT I·CLS
90 FOR I=1 TO LEN (A$)
100 B$=MID$(A$,I,1)
110 V=ASC(B$)-45
120 IF V<=32 THEN PRINT TAB(V)"
*" GOTO 30
130 V=V-26
```

```
140 PRINT TAB(V)"*"
150 NEXT I
```



```
5  HOME
10  PRINT  TAB( 9)"CODIGO DE DI
STANCIA": PRINT
20  PRINT  TAB( 14)"CUIDADO": P
RINT
30  PRINT  TAB( 4)"NAO DEIXE ES
PACO  ENTRE AS PALAVRAS"
40  PRINT : PRINT
50  PRINT "QUAL A SUA  MENSAGEM
 ": INPUT A$
60  FOR I = 1 TO 600: NEXT I: H
OME
70  FOR I = 1 TO  LEN (A$)
80 B$ =  MID$ (A$,I,1)
90 V =  ASC (B$) - 45
100  IF V <  = 32 THEN  PRINT
TAB( V)"*": GOTO 130
110 V = V - 26
120  PRINT  TAB( V)"*"
130  NEXT I
```

Inicialmente, o programa descobre o valor ASCII de cada letra de seu texto, através do comando **MID$** no laço que vai da linha 90 à linha 150 (130 no Apple e no TK-2000). Uma vez que a mensagem já foi convertida em uma série de números, é fácil fazer a codificação por meio de uma transformação linear. No caso do MSX, por exemplo, a letra que está na variável V será transformada na distância que é o seu valor em ASCII menos 26 (linha 130). O comando **TAB** imprimirá o asterisco na distância desejada, a partir do lado direito da tela (linha 120), e o processo de codificação estará pronto.

COMO USAR O CÓDIGO

Apesar de parecer demasiado simples para ser eficiente, o código de posição conta com vários fatores a seu favor. Em primeiro lugar, antes de decifrar um código, você deve verificar se ele realmente existe. Isso porque, sendo constituída de vários pontos (ou asteriscos) dispostos aparentemente de forma aleatória sobre qualquer superfície, uma mensagem assim codificada facilmente passaria despercebida. Durante a Segunda Guerra, aliás, esse truque foi usado por agentes de vários países. Assim, após uma inspeção mais cuidadosa na inocente fotografia de um quintal, descobriu-se que as estacas de um varal dispunham-se de maneira a formar uma mensagem secreta.

Um modo de fazer com que o código de posição fique mais difícil de ser quebrado é reestruturar o programa, tornando as letras-chave realmente aleatórias. Sem essa alteração, um perito que saiba estar manipulando um código de posição terá de tentar no máximo 26 combinações antes de resolver o problema. Se a ordem das letras for aleatória, o número de combinações crescerá extraordinariamente.



3

Cifra de Saint Cyr

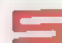
CIFRA DE SAINT CYR

Na Antiguidade, os maiores criptógrafos depois dos gregos foram os romanos. Júlio César, por exemplo, inventou um código no qual cada letra era substituída pela terceira letra que se lhe seguia no alfabeto. Nesse caso, o A torna-se D, o B torna-se E, e assim por diante. No fim do alfabeto, o X torna-se A, o Y, B e o Z é substituído pelo C. Usando esse método, a mensagem MINAR CAMPO LADO LESTE cifrada seria: PLQDU FDPSR ODGR OHVWH.

Como você verá mais tarde, o código criado por Júlio César é um caso especial da chamada cifra de Saint Cyr e você poderá verificar aquela mensagem rodando o próximo programa e usando 3333333 como número-chave.

Com o fim do Império Romano, a criptografia passou por longo período de estagnação e, apesar do crescente uso de mensagens cifradas nos séculos XVI e XVII, só no século XIX a Academia Militar Francesa de Saint Cyr produziu inovações sobre o código de César. A cifra de Saint Cyr é constituída de uma seqüência de letras em ordem alfabética, abaixo da qual colocamos um alfabeto, a partir de um ponto desejado. Nesse caso, cada letra do alfabeto de baixo terá sua correspondente na linha de cima. Assim, como mostra a ilustração desta página, a palavra INPUT (na linha de baixo) seria cifrada como AFHML. Uma das vantagens da cifra de Saint Cyr é que podemos escolher um ponto de partida diferente para cifrar cada letra, bastando que o receptor conheça esses pontos. Isto o torna ainda mais difícil de ser decifrado.

No programa *Cifra de Saint Cyr* incorporou-se a esse código um número-chave, a fim de conferir-lhe maior segurança. Mesmo que consiga ter acesso à listagem do programa, nenhum "espião" decifrará a mensagem, a menos que ele conheça o número secreto.



```
 20 BORDER 0: PAPER 0: INK 7:
CLS
 25 POKE 23658,8
 30 PRINT TAB 8;"CIFRA DE ST.C
YR": PRINT
 40 PRINT  INK 2; PAPER 7;
FLASH 1;AT 6,10;"  CUIDADO  "
 50 PRINT : PRINT "Nao deixe e
spacos entre palavras"
 60 PRINT : PRINT
 70 PRINT ''"  1 -> codificar"
 80 PRINT " -1 -> decodificar"
 90 INPUT s
100 INPUT "Qual a mensagem ?"'
a$
110 PAUSE 50: CLS
120 INPUT "Qual o numero-chave
? (7 digitos)"'n$
130 PAUSE 50: CLS
140 FOR k=1 TO LEN a$
150 LET l=k-INT (k/7)*7+1
160 LET t=CODE (a$(k))+(s*VAL
(n$(l)))
170 IF t>90 OR t<65 THEN  LET
t=t-(s*26)
180 PRINT CHR$ (t);
190 NEXT k
```



```
20 CLS
30 PRINT @7,"CIFRA DE ST. CYR"
40 PRINT @140,"CUIDADO":PRINT
50 PRINT"NAO DEIXE ESPACOS ENTR
E PALAVRAS"
60 PRINT:PRINT
70 PRINT" < 1> PARA CODIFICAR"
80 PRINT" <-1> PARA DECODIFICAR
"
90 INPUT S
100 INPUT"QUAL A MENSAGEM";A$
110 FOR K=1 TO 1500:NEXT:CLS
120 INPUT"QUAL O NUMERO CHAVE (
7 DIGITOS) ";N$
130 FOR K=1 TO 1500:NEXT CLS
140 FOR K=1 TO LEN(A$)
150 L=K-INT(K/7)*7+1
160 T=ASC(MID$(A$,K,1))+(S*VAL(
MID$(N$,L,1)))
170 IF T>90 OR T<65 THEN T=T-(S
*26)
180 PRINT CHR$(T);
190 NEXT
```



```
20 CLS
30 PRINT TAB(10)"CIFRA SAINT CY
R":PRINT
40 PRINT TAB(14)"CUIDADO" PRINT
50 PRINT TAB(2)"NAO DEIXE ESPAÇ
 O ENTRE AS PALAVRAS"
```

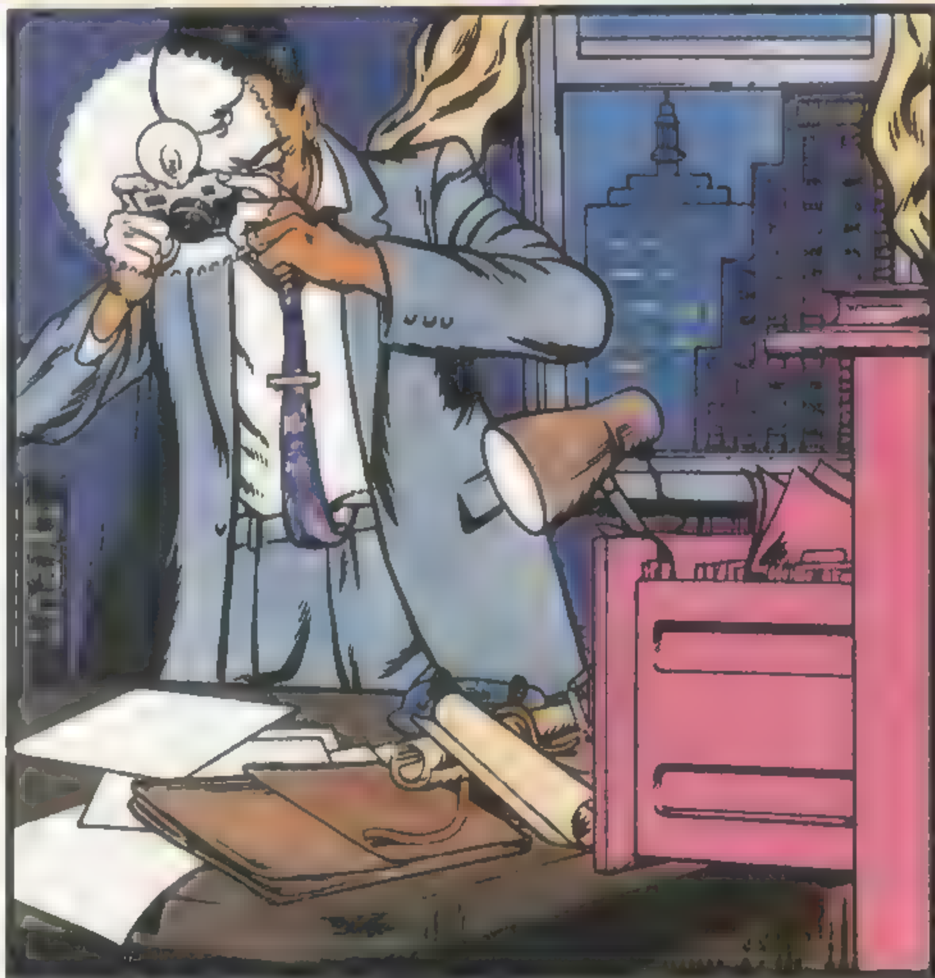
```
60 PRINT:PRINT
70 PRINT"DIGITE  1 PARA CODIFIC
AR"
80 PRINT"DIGITE -1 PARA DECODIF
ICAR"
90 INPUT S
100 INPUT"QUAL SUA MENSAGEM";A$
110 FOR K=1 TO 1500:NEXT K:CLS
120 INPUT"introduza o numero ch
ave (7 digitos)",N$
130 FOR K=1 TO 1500:NEXT K:CLS
140 FOR K=1 TO LEN(A$)
150 L=K-INT(K/7)*7+1
160 T=ASC(MID$(A$,K,1))+(S*VAL(
MID$(N$,L,1)))
170 IF T>90 OR T<65 THEN T=T-(S
*26)
180 PRINT CHR$(T);
190 NEXT
```



```
20  HOME
30  PRINT  TAB( 11)"CIFRA SGT
CYR": PRINT
40  PRINT  TAB( 14)"CUIDADO": P
RINT
50  PRINT "NAO DEIXE ESPACOS EN
TRE AS PALAVRAS"
60  PRINT : PRINT
70  PRINT "INTRODUZA  1  PARA C
ODIFICAR"
80  PRINT "INTRODUZA -1 PARA DE
CODIFICAR"
90  INPUT S
100  INPUT "QUAL SUA MENSAGEM?"
;A$
110  FOR K = 1 TO 1500: NEXT K:
 HOME
120  INPUT "INTRODUZA O NUMERO
CHAVE (7 DIGITOS)";N$
130  FOR K = 1 TO 1500: NEXT K
 HOME
140  FOR K = 1 TO  LEN (A$)
150 L = K -  INT (K / 7) * 7 +
1
160 T =  ASC ( MID$ (A$,K,1)) +
 (S *  VAL ( MID$ (N$,L,1)))
170  IF T > 90 OR T < 65 THEN T
 = T - (S * 26)
180  PRINT  CHR$ (T);
190  NEXT
```

Cada letra lida do texto do programa é, antes, convertida em seu valor ASCII. A função **VAL** acrescenta a esse valor uma quantidade indicada por um dos dígitos de seu número-chave (linha 160). Depois de conferirmos se o resultado está na faixa aceitável (linha 170) a função **CHR$** apresentará a letra equivalente à original.

Consideremos a mensagem TROPASCAPTURAMPONTEPEGASUS. Rodando o programa com o número-chave 5331401, o texto codificado seria: **WUPTATHDSUYRBRSROXEQJJDTYS.** Com um indicador variável **S**, que pode assumir os valores −1 ou 1, é possível usar o mesmo programa para a decodificação.



O número-chave de sete dígitos pode ser escolhido aleatoriamente. Porém, como é necessário que ele esteja sempre disponível, é aconselhável utilizar algo que nos seja familiar, como um número de telefone. Como há dez milhões de combinações possíveis para o número-chave, a cifra de Saint Cyr dificilmente pode ser quebrada. Contudo, é possível torná-la ainda mais segura se codificarmos a mensagem duas vezes, isto é, colocando a mensagem já cifrada outra vez no computador.

Usando os números-chave 8694153 e 8130337, a seqüência de codificação e decodificação seria:

Texto	Número-chave	
PREPARARDESEMBARQUE		CODIFICAÇÃO
	8694153	
VAIQFUIXMITJPJGAUVJ		
	8130337	
WDITFBQYPIWMWRHDUYM		DECODIFICAÇÃO
	8130337	
VAIQFUIXMITJPJCAUVJ		
	8694153	
PREPARARDESEMBARQUE		

CÓDIGO MORSE

A segurança, porém, não é tudo. Por mais indecifrável que seja um código, ele de nada vale se não pode ser transmitido rapidamente.

O imperador francês Napoleão Bonaparte percebeu o problema e tentou superá-lo construindo por toda a França torres que se comunicavam por meio de sinais de luz. Isso tornou possível o envio de mensagens a grandes distâncias em pouco tempo. Dessa forma, o Exército francês podia manter-se informado sobre o movimento das tropas inimigas.

Entretanto, a invenção do telégrafo e da cifra conhecida como código Morse, na década de 1830, foram os fatores que mais contribuíram para dinamizar as comunicações secretas. O inventor desse código, o norte-americano Samuel Morse, criou um sistema de comunicação no qual pontos e traços substituíam as letras.

O programa *Código Morse* codificará seus textos e decodificará uma série de pontos e traços. Por exemplo, o famoso sinal de socorro SOS seria repre

P&R

Qual a utilidade dos códigos?

Indispensáveis ao mundo dos espiões, os códigos são também muito empregados nas telecomunicações e em atividades bancárias que exigem sigilo. Grandes empresas comerciais que fazem uso de computadores para controlar seus estoques recorrem igualmente a esse tipo de linguagem. Assim, cada item estocado é classificado com um código. Quando se trata de artigos importados, costuma-se utilizar o "código de barras", que consiste em quadrinhos com faixas verticais pretas e brancas. Essas "barras" são lidas por meio de um instrumento óptico. Quando um dos artigos é retirado, o computador registra a operação e informa a respeito do preço da mercadoria.

Existem também terminais de compras nas próprias lojas, onde o cliente apenas digita um código e o total a pagar. Instantaneamente, o computador transfere essa quantia da sua conta para a da loja. Como se pode ver, as notas e moedas tendem a desaparecer, e as caixas registradoras deverão ser substituídas por terminais de computador. Mas, para tanto, é necessário que sejam criados códigos muito eficientes, de modo a evitar roubos e fraudes.

sentado por /.../—/.../ e um texto maior como FUJA À MEIA-NOITE seria /..-./..-/.---/.-/.-/--/./ ./.-/-./---/../-/./

UM EXERCÍCIO DE CODIFICAÇÃO

Tente fazer as duas codificações mencionadas com o próximo programa. Para os micros da linha Sinclair, você deve adicionar cinco asteriscos no final da mensagem para indicar que ela já está completa.

S

```
10 BORDER 0· PAPER 0: INK 7·
CLS · DIM a$(26,4): LET S$="
"· LET f$="      "
 15 POKE 23658,8
 20 FOR x=) TO 26· READ a$(x):
 NEXT x
 30 INPUT "Codificar (1) ou De
codificar (2)";r
 40 IF r=2 THEN  GOTO 140
 60 INPUT "MENSAGEM A SER CODI
FICADA  "'m$
 70 FOR x=1 TO LEN m$
 80 IF m$(x)=" " THEN  PRINT "
    ";· GOTO 110
 90 LET p$=m$(x)
100 PRINT "     ".a$((CODE p$)-
64);
110 NEXT x
120 PRINT ''''" QUALQUER TECLA
 PARA CONTINUAR";· PAUSE 9999
130 RUN
140 INPUT "MENSAGEM A SER DECO
DIFICADA :"'m$· LET m$=m$+" "
160 FOR x=1 TO LEN m$
170 LET k$=m$(x)
180 IF k$=" " THEN  GOTO 220
190 LET s$=s$+k$
200 NEXT x: GOTO 120
210 IF LEN s$>5 OR LEN s$<1
THEN  PRINT "ERRO"  GOTO 120
220 IF LEN s$<>5 THEN  LET s$=
s$+f$( TO 5-LEN s$)
225 FOR h=1 TO 26· IF a$(h)=s$
THEN  PRINT CHR$ (h+64).
230 NEXT h
240 LET s$="": GOTO 200
250 DATA "0-","-000","-0-0","-
00","0","00-0","--0","0000","0
0","0---","-0-","0-00","--","
0","---","0--0","--0-","0-0","
000","-","00-","000-","0- ","-
00-","-0--","--00"
```

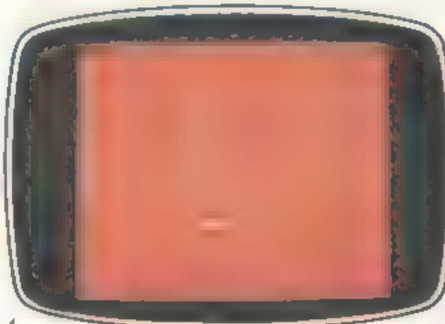
```
10 CLS:DIM A$(26)
20 FOR X=1 TO 26:READ A$(X):NEX
T X
30 PRINT @96,;:INPUT"CODIFICAR
OU DECODIFICAR (1,2) ";R
40 IF R=2 THEN 140
50 PRINT"  MENSAGEM A SER CODIF
ICADA  :"
60 INPUT M$
70 FOR X=1 TO LEN(M$)
80 IF MID$(M$,X.1)=" " THEN PRI
NT"     ";:GOTO 110
90 P$=MID$(M$,X,1)
100 PRINT " ";A$(ASC(P$)-64);
110 NEXT
120 PRINT·PRINT:PRINT "  QUALQU
ER TECLA PARA CONTINUAR"
130 IF INKEY$="" THEN 130 ELSE
RUN
140 PRINT" MENSAGEM A SER DECOD
IFICADA"
150 INPUT M$:M$=M$+" ":PRINT:PR
INT
160 FOR X=1 TO LEN(M$)
170 K$=MID$(M$.X.1)
180 IF K$=" " THEN 210
190 S$=S$+K$
200 NEXT:PRINT GOTO 120
210 IF LEN (S$)>4 OR LEN(S$)<1
THEN PRINT " ";·GOTO 200
220 FOR H=1 TO 26·IF A$(H)=S$ T
HEN PRINT CHR$(H+64);
230 NEXT
240 S$="":GOTO 200
250 DATA 0-,-000,-0-0,-00,0,00-
0,--0,0000,00,0---,-0-,0-00,--,
-0,---,0--0,--0-,0-0,000,-,00-,
000-,0--,-00-,-0--,--00
```

```
10 CLS·DIM A$(26)
20 FOR X=1 TO 26  READ A$(X)·NE
XT X
30 PRINT TAB(3);:INPUT"CODIFICA
```



```
R OU DECODIFICAR(1,2)";R
40 IF R=2 THEN 140
50 PRINT TAB(7)"MENSAGEM A SER
CODIFICADA"
60 INPUT M$
70 FOR X=1 TO LEN (M$)
80 IF MID$(M$,X,1)=" " THEN PRI
NT"     ",·GOTO 110
90 P$ MID$(M$,X,1)
100 PRINT" ";A$(ASC(P$)-64);
110 NEXT X
120 PRINT·PRINT:PRINT TAB(2)"AP
ERTE QUALQUER TECLA PARA CONTIN
UAR"
130 IF INKEY$="" THEN 130 ELSE
RUN
140 PRINT TAB(6)"MENSAGEM A SER
 DECODIFICADA"
150 INPUT M$ M$=M$+" "·PRINT·PR
INT
160 FOR X=1 TO LEN(M$)
170 K$=MID$(M$,X,1)
180 IF K$=" " THEN GOTO 210
190 S$ S$+K$
200 NEXT X·PRINT·GOTO 120
210 IF LEN(S$)>4 OR LEN(S$)<1 T
HEN PRINT:PRINT "    DEIXE ESPA
ÇO ENTRE OS CÓDIGOS": GOTO 200
220 FOR H=1 TO 26 IF A$.H)=S$ T
HEN PRINT CHR$(H+64):
230 NEXT
240 S$ ""·GOTO 200
250 DATA 0-,-000,-0-0,-00,0,00-
0,--0,0000,00,0---, 0 ,0 00, -,
 0,    ,0 -0,--0-,0 0,000,-,00 ,
000-,0--,-00-,-0--,--00
```

```
10  HOME : DIM A$(26)
20  FOR X = 1 TO 26: READ A$(X)
: NEXT X
30  PRINT  TAB( 6);: INPUT "COD
IFICAR OU DECODIFICAR(1,2)?";R
```



4

Código Morse na tela

```
40  IF R = 2 THEN 140
50  PRINT  TAB( 9)"MENSAGEM A S
ER CODIFICADA"
60  INPUT M$
70  FOR X = 1 TO  LEN (M$)
80  IF  MID$ (M$,X,1) = " " THE
N  PRINT "     ";· GOTO 110
90 P$ =  MID$ (M$,X,1)
100  PRINT " ";A$( ASC (P$) - 6
4);
110  NEXT
120  PRINT   PRINT   PRINT  TAB
( 3)"APERTE QUALQUER TECLA PARA
 CONTINUAR"
130  GET T$· IF T$ = "" THEN  G
OTO 130
135  GOTO 10
140  PRINT  TAB( 8)"MENSAGEM A
SER DECODIFICADA"
150  INPUT M$:M$ = M$ + " "; PR
INT · PRINT
160  FOR X = 1 TO  LEN (M$)
170 K$ =  MID$ (M$,X,1)
180  IF K$ = " " THEN  GOTO 210
190 S$ = S$ + K$
200  NEXT X: PRINT : GOTO 120
210  IF  LEN (S$) > 4 OR  LEN (
S$) < 1 THEN  PRINT " ",  GOTO
200
220  FOR H = 1 TO 26  IF A$(H)
= S$ THEN  PRINT  CHR$ (H + 64)
;
230  NEXT
240 S$ = "": GOTO 200
250  DATA  0-,-000,-0-0,-00,0,0
0-0,--0,0000,00,0---,-0-,0-00,-
-,-0,---,0--0,--0-,0-0,000,-,00
-,000-,0--,-00-,-0--,--00
```

A primeira parte do programa atribui a cada variável **A$(X)** um sinal Morse equivalente a uma letra do alfabeto, na sua devida ordem (linha 20). Por exemplo, **A$(1)** conterá /.-/, que é o código da letra A. Seria interessante usar o sinal de menos para um traço, e o asterisco ou o zero para um ponto, em vez do ponto final.

A parte de codificação é muito semelhante à dos dois primeiros programas. Cada letra de seu texto é lida e convertida em um número entre 1 e 26 e a sequência adequada de pontos e traços é então impressa (linha 100).

A decodificação da mensagem se processa do seguinte modo. O computador lê cada caractere e soma-o aos anteriores, até encontrar um espaço (linhas 160 a 200). Então ele compara essa cadeia de caracteres com as que estão na variável **A$**, identificando a letra equivalente e imprimindo-a por meio do comando **CHR$** (linha 220).

Na segunda parte deste artigo, você aprenderá a decodificar transposições e cifras e utilizar códigos multiplicativos e códigos comerciais.

ARMAZENAGEM DE NÚMEROS

Como você já sabe, o interpretador BASIC pode trabalhar com números fracionários. Aprenda agora a armazená-los na memória e obtenha resultados mais precisos.

Quando o resultado de uma operação numérica é muito grande ou, então, muito pequeno, o computador mostra, freqüentemente, na tela, um número com aparência meio estranha, tal como 2.34E12 ou 1.222E-9.

Essa maneira de representar números é conhecida como *notação científica*, ou *notação de engenharia*, e é usada automaticamente toda vez que o interpretador BASIC não consegue mostrar um valor numérico que caia dentro do limite de sete a oito algarismos, dependendo da marca do computador. Existem também meios, em BASIC, de mostrar ou imprimir os valores numéricos que se quiser, em notação científica.

Na notação científica (chamada ainda de *forma exponencial*), um número é composto por duas partes: a *mantissa*, que contém os algarismos significativos do número, geralmente com apenas um dígito à esquerda da vírgula; e o *expoente*, que consiste em uma potência de 10 pela qual a mantissa deverá ser multiplicada para obter o valor numérico a ser representado. Assim, por exemplo, os números aqui abordados significam, respectivamente:

$2.34E12 = 2.34 \times 10^{12} =$
$= 2.340.000.000\,000$

$1.223E\text{-}9 = 1.223 \times 10^{-9} =$
$= 0,000000001223$

Na realidade, esta é exatamente a maneira com que o interpretador BASIC armazena *todos* os valores reais: mantissa e expoente são conservados em grupos separados de bytes. Só quando o número vai ser mostrado por um **PRINT** ou equivalente é que se processa a conversão para o formato mais conhecido. O limite para que isso ocorra varia com a marca do computador. Nos micros da linha Apple, TRS-Color e TRS-80, por exemplo, é de sete algarismos. No Spectrum e no MSX, e de oito algarismos. Alguns computadores, como os das linhas TRS e MSX, podem expressar números com precisão dupla, isto é, não precisam recorrer à forma exponencial se o número tiver até quinze dígitos de precisão.

Você também poderá usar a forma exponencial como um modo abreviado de entrar números longos no computador (dentro de um programa, ou mesmo usando um comando **INPUT**).

O programa a seguir o ajudará a entender como funciona essa forma, e como o valor da mantissa é calculado.

S

```
10 CLS : POKE 23658,8
20 INPUT "INTRODUZA O NUMERO"
, LINE A$
25 IF A$="" THEN  GOTO 20
30 LET N$="0": LET E=0: LET N
```



- EXPOENTES
- COMO SÃO ARMAZENADOS OS NÚMEROS
- PONTO FLUTUANTE
- A FUNÇÃO INSTR
- NÚMEROS NEGATIVOS
- PEEK NA MEMÓRIA
- COMO ECONOMIZAR MEMÓRIA
- EFEITOS ESTRANHOS
- FORMATAÇÃO COM PRINT USING

```
=VAL A$
40 IF N=0 THEN PRINT "VALOR MUITO PEQUENO !": PAUSE 50: GOTO 10
50 LET F=0: FOR M=1 TO LEN A$ : IF A$(M)="E" THEN LET F=M
55 NEXT M: IF F=0 THEN GOTO 200
60 LET N$=STR$ N: LET F=0: FOR M=1 TO LEN N$: IF N$(M)="E" THEN LET F=M
65 NEXT M: IF F=0 THEN GOTO 180
68 LET N$=N$( TO F-1)
70 IF ABS N<1 THEN GOTO 130
80 IF ABS N<10 THEN GOTO 110
90 LET N=N/10: LET F=0: FOR M=1 TO LEN N$: IF N$(M)="." THEN LET F=M
92 NEXT M: IF F=LEN N$ THEN LET N$=N$( TO LEN N$-1): GOTO 100
95 IF F<>0 THEN LET N$=N$( TO F-1)+N$(F+1)+"."+N$(F+2 TO ): GOTO 80
100 LET N$=N$+"0": GOTO 80
110 IF N$(LEN N$)="." THEN LET N$=N$( TO LEN N$-1)
120 GOTO 180
130 IF ABS N>=1 THEN GOTO 170
140 LET N=N*10: LET F=0: FOR M=1 TO LEN N$: IF N$(M)="." THEN LET F=M
145 NEXT M: IF F=1 THEN LET N$=" 0"+N$(2 TO ): GOTO 130
150 IF F<>0 THEN LET N$=N$( TO F-2)+"."+N$(F-1)+N$(F+1 TO ): GOTO 130
160 LET N$=" "+N$: GOTO 130
170 IF VAL A$<0 THEN LET N$="-"+N$
180 PRINT : PRINT A$;" IGUAL": PRINT N$
190 GOTO 20
200 IF ABS N<1 THEN GOTO 220
210 IF ABS N>=10 THEN LET E=E+1: LET N=N/10: GOTO 210
215 GOTO 230
220 IF ABS N<=1 THEN LET E=E-1: LET N=N*10: GOTO 220
230 PRINT A$;" IGUAL": PRINT N;: IF E<>0 THEN PRINT "E";E
240 PRINT : GOTO 20
```



```
10 CLS
20 PRINT:INPUT"INTRODUZA O NUMERO ";A$
30 N$="0":E=0:N=VAL(A$)
40 IF N=0 THEN PRINT "VALOR MUITO PEQUENO!":PRINT:GOTO 20
50 I=INSTR(A$,"E"):IF F=0 THEN 200
60 N$=STR$(N):F=INSTR(N$,"E"):IF F=0 THEN 180 ELSE N$=MID$(N$,2,F-2)
70 IF ABS(N)<1 THEN 130
80 IF ABS(N)<10 THEN 110
90 N=N/10:F=INSTR(N$,"."):IF F=LEN(N$) THEN N$=LEFT$(N$,LEN(N$)-1) ELSE IF F<>0 THEN N$=LEFT$(N$,F-1)+MID$(N$,F+1,1)+"."+MID$(N$,F+2):GOTO 80
100 N$=N$+"0":GOTO 80
110 IF RIGHT$(N$,1)="." THEN N$=LEFT$(N$,LEN(N$)-1)
120 GOTO 180
130 IF ABS(N)>=1 THEN 170
140 N=N*10:F=INSTR(N$,"."):IF F=1 THEN N$=".0"+MID$(N$,2):GOTO 130
150 IF F<>0 THEN N$=LEFT$(N$,F-2)+"."+MID$(N$,F-1,1)+MID$(N$,F+1):GOTO 130
160 N$="."+N$:GOTO 130
170 IF VAL(A$)<0 THEN N$="-"+N$
180 PRINT:PRINT A$;" IGUAL":PRINT N$
190 GOTO 20
200 IF ABS(N)<1 THEN 220
210 IF ABS(N)>=10 THEN E=E+1:N=N/10:GOTO 210 ELSE 230
220 IF ABS(N)<=1 THEN E=E-1:N=N*10:GOTO 220
230 PRINT A$;" IGUAL ":PRINT N;CHR$(8);:IF E<>0 THEN PRINT "E";E
240 GOTO 20
```



```
10  HOME
20  PRINT : INPUT "ENTRE NUMERO :";A$
30  LET N$ = "0":E = 0:N = VAL (A$)
40  IF N = 0 THEN PRINT "VALOR MUITO PEQUENO": PRINT : GOTO 20
50  FOR F = 1 TO LEN (A$): IF MID$ (A$,F,1) = "E" THEN 54
52  NEXT F:F = 0: GOTO 200
54  IF F = 0 THEN 200
60  LET N$ = STR$ (N)
62  FOR F = 1 TO LEN (N$): IF MID$ (N$,F,1) = "E" THEN 66
64  NEXT F:F = 0
66  IF F = 0 THEN 180
68  LET N$ = MID$ (N$,1,F - 2)
70  IF ABS (N) < 1 THEN 130
80  IF ABS (N) < 10 THEN 110
90  LET N = N / 10
92  FOR F = 1 TO LEN (N$): IF MID$ (N$,F,1) = "." THEN 96
94  NEXT F:F = 0
96  IF F = LEN (N$) THEN N$ = LEFT$ (N$, LEN (N$) - 1): GOTO 100
98  IF F < > 0 THEN N$ = LEFT$ (N$,F - 1) + MID$ (N$,F + 1,1) + "." + MID$ (N$,F + 2): GO
```

ARMAZENAMENTO DE NUMEROS

COMO CONVERTER

COMO MOVIMENTAR O PONTO



é mover o ponto até que ele fique à esquerda do primeiro 1, e adicionar 1 ao expoente para cada posição que o ponto seja deslocado. Por exemplo, para o número 58 (em decimal), ou 111010.000 (em binário), o ponto binário é movido gradualmente para a esquerda, até que não haja mais números à sua esquerda. Neste caso, teríamos que movê-lo seis casas para a esquerda, de modo que o expoente resultasse em +6, e a mantissa fosse 0.11101000.

Na realidade, o expoente parte de 128 (por razões esclarecidas mais adiante); portanto, o número de posições deslocadas deve ser somado a 128, e não a 0. No exemplo acima, o expoente será igual a 128+6, ou 134 (e isto é armazenado como um número binário).

Recapitulando, o computador armazena a parte exponencial de um número em byte. A mantissa é armazenada também, só que ocupa quatro bytes. No exemplo acima, os três bytes restantes seriam preenchidos com zeros:

EXPOENTE

MANT-1 MANT-2 MANT-3 MANT-4

11101000 00000000 00000000 00000000

Portanto, qualquer número real ocupa exatamente cinco bytes. Isso limita os tamanhos máximo e mínimo que o computador pode armazenar. O byte único da parte exponencial nos dá o valor máximo de 2 elevado a 127 (não de 256, pois o primeiro bit é usado como indicador de sinal, para comunicar se o expoente é positivo ou negativo, deixando apenas sete bits para o valor numérico do expoente).

O valor máximo da mantissa é 0.11111..., quando todos os bits são iguais a 1. Esse valor está muito próximo do binário 1.00000..., que é 1 tanto em binário quanto em decimal. O menor valor possível será, então, 0.100000..., quando todos os bits, exceto o primeiro, forem iguais a zero (lembre-se de que o primeiro bit da mantissa é sempre igual a 1, pois o ponto binário é movido até aí). E 0.1 em binário é igual a 1/2 ou 0.5, em decimal.

Multiplicando a mantissa pelo expoente, você poderá chegar até os valores extremos que podem ser armazenados na memória, em ponto flutuante. O valor máximo é 1.70141E38, em decimal, para um micro de oito bits. Você poderá utilizar os programas seguintes para verificar como é representado este número internamente.

Nesse aspecto, o Spectrum difere dos demais computadores: ele utiliza seis bytes em vez de cinco para armazenar números em ponto flutuante. Cinco desses bytes são idênticos aos do exemplo citado. O sexto é necessário para indicar ao computador se o número é de ponto flutuante ou não. Isso é feito precedendo-se o grupo de cinco bytes com o byte igual a 14.

CASOS ESPECIAIS

O que foi dito anteriormente nos dá uma boa idéia de como o computador armazena números em formato de ponto flutuante. Existem, porém, duas variantes em relação ao processo estudado. A primeira ocorre quando o número inicial é menor do que 1. Nesse caso, se tentarmos movimentar o ponto binário para a esquerda, iremos afastá-lo do lugar onde queremos que ele realmente esteja. Portanto, devemos movê-lo para a direita. Do mesmo modo, a fim de que o número seja corretamente representado, diminui-se o expoente de 1, para cada posição deslocada (em vez de somar 1). O expoente começa, como sempre, a partir do valor 128.

Muitas vezes, antes de exibir um número binário, o computador escreve, acima de cada "coluna" desse número, seu equivalente decimal. Você já deve ter visto isso antes nos artigos sobre blocos gráficos, mas apenas para o caso dos números à esquerda do ponto. O mesmo pode ser feito para os números à direita do ponto, que formam na realidade sua parte fracionária.

Em binário, para achar o valor decimal da próxima coluna à esquerda, multiplica-se o valor por 2. Trabalhando da esquerda para a direita, portanto, deve-se fazer exatamente o contrário, ou seja, dividir por 2.

A segunda variante na representação dos números em ponto flutuante ocorre quando o número é menor do que zero, ou seja, negativo. Nesse caso, o computador precisa armazenar a sua versão interna do sinal de menos em algum lugar — e isto deve ser feito sem desperdiçar espaço de memória.

Você viu antes que, da maneira como os números são armazenados, o primeiro bit do primeiro byte da mantissa tem que ser sempre igual a 1. Dando is-

so como certo, o computador utiliza esse bit para indicar o sinal da mantissa: se o número for negativo, o bit valerá 1; se ele for positivo, o valor do bit será, então, 0.

ONDE ESTÁ AQUELE NÚMERO?

A partir dos exemplos acima, já é possível entender como o computador armazena os números na memória. O número decimal 58 (111010 em binário) é armazenado em cinco bytes sucessivos, da maneira como se segue:

134 104 000 000 000 (em decimal)

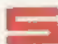
e o número 10 (decimal) é armazenado assim:

132 032 000 000 000

Experimente vários exemplos, trabalhando com os valores decimais contidos nos bytes. Teste as suas respostas usando o programa a seguir. Digite **RUN 100** para entrar outro número.

Embora armazenem números na forma geral exposta até agora, os micros compatíveis com o Sinclair Spectrum contam ainda com outro recurso. Se um número inteiro, compreendido entre -65535 e 65535 for usado, o Spectrum o armazena de modo diferente, reduzindo-o a seu equivalente binário e guardando-o em apenas dois bytes (e não em forma exponencial, usando seis bytes). Embora possa parecer que isso seja feito para economizar memória, tal fato não acontece, pois três bytes são deixados sem uso.

Por essa razão, quando você usar o primeiro dos dois programas abaixo, não entre um valor númerico entre -65535 e 65535: o programa ainda funcionará, mas os resultados não corresponderão à explicação adotada.



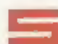
```
10 BORDER 1: PAPER 7: INK 9:
CLS
20 INPUT "Introduza um numero
(nao inteiro)",x
40 PRINT '"Expoente: ";PEEK (
PEEK 23627+256*PEEK 23628+1)
50 PRINT '"Mantissa:";: FOR n
=2 TO 5
60 PRINT TAB 10;PEEK (PEEK
23627+256*PEEK 23628+n)
70 NEXT n
80 STOP
```



```
10 CLS
20 INPUT" INTRODUZA UM NUMERO "
;N
30 D=VARPTR(N)
40 PRINT:PRINT" EXPOENTE = ",PE
EK(D)
50 PRINT" MANTISSA = ";
60 FOR G=1 TO 4
70 PRINT,PEEK(D+G)
80 NEXT
90 PRINT:GOTO 20
```



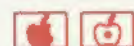
```
10  HOME
20  INPUT "ENTRE UM NUMERO:";X
30  LET V =  PEEK (105) +  PEEK
 (106) * 256
40  PRINT "EXPOENTE:"; PEEK (V
+ 2)
50  PRINT "MANTISSA:";
60  FOR N = 3 TO 6
70  PRINT  PEEK (V + N);" ";
80  NEXT N
```



```
100 REM SEGUNDO PROGRAMA
110 BORDER 1: PAPER 7: INK 9:
CLS : LET r=0
120 INPUT AT 1,0;"Introduza ex
poente",exp
130 IF exp<0 OR exp>255 THEN
GOTO 120
140 PRINT '"Expoente: ";exp:
POKE PEEK 23627+256*PEEK 23628
+1,exp
150 PRINT '"Mantissa:";: FOR n
=2 TO 5
160 INPUT AT 1,0;"Introduza ma
ntissa",man
170 IF man<0 OR man>255 THEN
GOTO 160
180 PRINT TAB 10;man: POKE
PEEK 23627+256*PEEK 23628+n,
man
190 NEXT n
200 PRINT '"Resulta: ";r
```



```
100 REM SEGUNDO PROGRAMA
110 CLS
120 N=1:D=VARPTR(N)
130 INPUT" INTRODUZA EXPOENTE "
;E
140 POKE D,(255 AND E)
150 PRINT" INTRODUZA MANTISSA "
;
160 FOR K=1 TO 4
170 INPUT E
180 POKE D+K,(255 AND E)
190 PRINT ,;
200 NEXT
210 PRINT:PRINT" NUMERO E ";N:P
RINT
220 GOTO 130
```



```
110  HOME :R = 1:V =  PEEK (105
) +  PEEK (106) * 256
120  INPUT "ENTRE EXPOENTE:";EX
130  IF EX < 0 OR EX > 255 THEN
 120
140  POKE V + 2,EX
150  FOR N = 3 TO 6
160  INPUT "ENTRE MANTISSA:";M
170  IF M < 0 OR M > 255 THEN 1
60
180  POKE V + N,M
190  NEXT N
200  PRINT "RESULTADO=";R
```

O primeiro desses programas permite que um número seja digitado pelo teclado; em seguida, ele imprime os conteúdos decimais dos cinco bytes onde o número está guardado.

Seu funcionamento tem a seguinte dinâmica: inicialmente, é necessário estabelecer uma variável numérica, que armazenará o número de entrada. Como ela é a primeira variável a ser definida quando o programa é executado, seu nome aparecerá logo no começo de uma área especial da memória (definida e fixada pelo interpretador BASIC de cada computador), chamada lista de variáveis. Essa lista tem informações sobre o nome da variável e o endereço da memória RAM onde seu conteúdo será armazenado. Tal endereço é dividido em dois bytes sucessivos. No Apple, por exemplo, o endereço do conteúdo da primeira variável de um programa encontra-se nos apontadores 105 e 106 (em decimal). No Spectrum, os endereços correspondentes são 23627 e 23628. Tanto no Apple quanto no Spectrum, o endereço decimal de dezesseis bits é calculado pela expressão na linha 40 (30 no Apple).

Nos computadores das linhas TRS-80, TRS-Color e MSX, o usuário não precisa conhecer a locação exata do apontador da lista de variáveis, pois existe uma função predefinida do BASIC chamada **VARPTR** (abreviatura de *VARiable PoinTeR* — ou apontador de variáveis), que indica o endereço decimal completo do ponto da memória onde está armazenado o conteúdo da variável nomeada no argumento da função. Veja a linha 30 do programa para esses computadores. A variável **D** conterá o endereço inicial dos cinco bytes que contém o valor armazenado na variável **N**. O primeiro byte contém o expoente e nos quatro bytes sucessivos está a mantissa.

O segundo programa faz o contrário: pede que cinco bytes sejam digitados, e imprime o número que ele representa (algumas das respostas poderão aparecer na forma exponencial).

Seu funcionamento é semelhante ao do primeiro programa, ou seja, comandos **PEEK** (nas versões para o Apple e Spectrum) ou **VARPTR** (nas versões para o TRS e o MSX) são usados para localizar o endereço inicial da primeira va-

riável. Ela é igualada a 0 ou a 1 logo na primeira linha do programa, para assegurar que ficará em primeiro lugar na lista de variáveis (esse "truque" só é necessário para o Apple e o Spectrum). A seguir, o programa pede ao usuário que entre os valores do expoente e das partes da mantissa. Em vez de empreender cálculos infindáveis para chegar ao resultado convertido, o programa coloca esses valores nos cinco bytes de armazenamento da variável inicial, por meio de comandos **POKE**, e imprime o resultado, por meio de um **PRINT** normal.

ECONOMIZE MEMÓRIA

O computador sempre armazena números em grupos de cinco bytes. Ora, isso representa um grande desperdício de memória, pois nem sempre são necessários tantos bytes na mantissa. Além disso, os bytes em excesso poderiam ser aproveitados em outros lugares.

De fato, você pode economizar quantidades significativas de memória se evitar o uso de *números* em BASIC, embora isso nem sempre seja possível. Existe, entretanto, um jeito de reduzir a quantidade de memória que cada número requer. Assim, é possível, em muitos casos, evitar o emprego de números em um programa, colocando dentro de variáveis os números repetidamente utilizados em várias partes do programa. Por exemplo, uma constante que aparece com freqüência , como o número *pi* (3.1415...), pode ser colocado na variável **P**. Esse "truque" funciona sempre, mas não vale a pena recorrer a ele quando o valor numérico é usado apenas uma ou duas vezes no programa.

A vantagem de se empregar uma variável nesses casos é que o seu nome ocupa apenas um ou dois bytes de espaço no programa, em vez de cinco, se seu nome for curto. (A vantagem deixará, naturalmente, de existir se a variável se chamar **MEDIA**, ou coisa parecida).

Alguns números — como 0, 1, 2, 10 etc. — aparecem tantas vezes na maioria dos programas, que talvez convenha colocá-los em variáveis, se você tem problema de espaço. Esse método é especialmente útil nos micros Sinclair, que usam seis bytes em vez de cinco.

EFEITOS ESTRANHOS

Cálculos realizados por computador revelam, às vezes, erros aparentemente inexplicáveis; assim, um resultado que deveria dar "redondo" — 10.000000, por exemplo —, pode aparecer na tela sob uma forma levemente modificada — como 10.000001. Conhecidos como *erros de precisão*, esses equívocos são facilmente explicados pela maneira como o computador armazena números.

Tente os exemplos seguintes em sua máquina. A causa por trás de cada um dos erros será explicada mais adiante:

Primeiro, tente a linha abaixo:

```
PRINT 10.0000000001
```

Ao imprimir o resultado, o computador omite o 1 final e converte o número que você digitou em 10, simplesmente. Isso pode se tornar um problema, quando você quiser trabalhar com maior precisão numérica. Entretanto, é possível reverter esse fato em seu favor, quando se deseja entrar números entre -65535 e 65535 no programa (o que normalmente não deve ser feito, pois o Spectrum os representa de forma diferente, como já foi dito).

Basta, para isso, entrar o número inteiro em forma fracionária e colocar um algarismo pouco significativo no final. Assim, entrando o número 10.0000000001, como no exemplo citado, você "enganará" o microcomputador e ele mostrará como 10 é armazenado no formato de cinco bytes, em vez de no formato especial para números inteiros.

Agora tente entrar esta linha:

```
PRINT INT -65536
```

Com ela, será evidenciado um erro no interpretador BASIC do Spectrum, uma vez que ele não consegue diferenciar entre esse valor e -1.

Você pode testemunhar os estranhos efeitos causados nos micros compatíveis com o TRS-Color, entrando os números abaixo no primeiro programa:

```
12345678901 E-4
```

e também:

```
454545454 E-46
```

A discrepância do primeiro número é bastante óbvia. No segundo, o último dígito é mudado pelo computador.

Essas imprecisões não são, contudo, como muitas pessoas imaginam, erros na ROM dos computadores (exceção feita ao segundo exemplo para o Spectrum). A razão pela qual o algarismo 1 aparece ou desaparece, quando colocado no final do número, decorre do fato de que os computadores digitais, pela sua própria natureza, trabalham com limites no grau de precisão. Portanto, quando os valores excederem um determinado número de decimais, i interpretador será obrigado a arredondar o resultado.

Quando isso acontece, dependendo dos números que estão sendo usados e, também, do número de etapas de cálculo que o computador terá que vencer, os resultados finais podem ser completamente distintos do que se espera.

Os computadores são mais do que precisos a maior parte das vezes, e esses "erros", na realidade, não têm grande influência. Mas, para aplicações mais sérias, e que exigem grande precisão, como programas de contabilidade, de estatística etc., as incorreções cumulativas podem se tornar um grande inconveniente; isso é levado em consideração pelos melhores pacotes de software.

Por exemplo, usando conjuntos e matrizes, podem-se armazenar números com mais decimais do que o máximo de sete a nove permitido pelos computadores descritos no artigo. A vantagem dessa abordagem é que o programador pode desenvolver uma rotina para imprimir os números em forma normal, em vez da exponencial, como o computador faria com valores extremos. Essa é uma maneira, também, de evitar mensagens de erro, como "*overflow*", "número muito grande" etc.

FORMATAÇÃO DE NÚMEROS

Já explicamos que alguns microcomputadores, como o TRS-80, o TRS-Color e o MSX, têm comandos que permitem mudar o formato de impressão de números com o **PRINT** ou o **LPRINT**.

Embora uma tela possa ser formatada muito bem com o auxílio dos comandos **PRINT@** (nos TRS) ou **LOCATE** (no MSX), é conveniente recorrer à declaração **USING** para formatação, que é sempre usada em conjunto com os comandos **PRINT** ou **LPRINT**. Os elementos essenciais dessa declaração já foram explicados em artigo anterior; assim, faremos aqui apenas uma nova exposição, por intermédio de um programa simples de demonstração financeira para uma companhia fictícia.

```
10 CLS
20 PRINT"   A DIRETORIA TEM O PR
AZER DE    ANUNCIAR O BALANCO P
```

```
RELIMINAR              DE 1986 DA
:"
30 PRINT:PRINT"   FABRICA ACME
DE ARRUELAS   "
40 PRINT:PRINT:PRINT"O MAIOR FO
RNECEDOR DO MUNDO DE":PRINT
50 C$(0)="MD.345 - AZUIS":C$(1)
="MD.897 - VERDES":C$(2)="MD.91
2 - AMARELAS":C$(3)="MD.989 - V
ERMELHAS"
60 FOR K=0 TO 3
65 X$="ARRUELAS "+MID$(C$(K),10
)
70 PRINT TAB(16-LEN(X$)/2);X$
80 NEXT
90 FOR K=1 TO 7000:NEXT:CLS
110 PRINT @12,"JANEIRO":PRINT @
44,"-------"
120 PRINT:PRINT"PRECO MEDIO (AT
ACADO)"
130 A$="  %      %":B$="        *
*$##.##"
140 PRINT:PRINTUSING A$;C$(0);:
PRINTUSING B$;12.715265
150 PRINTUSING A$;C$(1);:PRINTU
SING B$;3.7363141
160 PRINTUSING A$;C$(2);:PRINTU
SING B$;10.35824221
170 PRINTUSING A$;C$(3);:PRINTU
SING B$;.5163733
180 PRINT @416,USING"LUCRO TOTA
L      $###,###.##";374241.5353
```

Esse programa, tal como está, funcionará bem apenas no micro TRS-Color. Para adaptá-lo para o TRS-80, faça as seguintes alterações:

```
70 PRINT TAB(32-LEN(C$(K)));C$(
K)
90 FOR K= 1 TO 4000:NEXT:CLS
110 PRINT @12,"JANEIRO": PRINT
@76, "-------"
180 PRINT @832,USING"LUCRO BRU-
TO     $###,###.##";374241.5353
```



```
10 SCREEN1:WIDTH32:CLS
20 PRINT"  A DIRETORIA TEM O PR
AZER DE    ANUNCIAR O BALANCO P
RELIMINAR              DE 1986 DA
:"
30 PRINT:PRINT"   FABRICA ACME
DE ARRUELAS   "
40 PRINT:PRINT:PRINT"O MAIOR FO
RNECEDOR DO MUNDO DE":PRINT
50 C$(0)="MD.345 - AZUIS":C$(1)
="MD.897 - VERDES":C$(2)="MD.91
2 - AMARELAS":C$(3)="MD.989 - V
ERMELHAS"
60 FOR K=0 TO 3
65 X$="ARRUELAS "+MID$(C$(K),10
)
70 PRINT TAB(16-LEN(X$)/2);X$
80 NEXT
90 FOR K=1 TO 7000:NEXT:CLS
110 PRINT TAB(12);"JANEIRO":PRI
NT TAB(12);"======="
120 PRINT:PRINT"PRECO MEDIO (AT
ACADO)"
130 A$="  /      /":B$="        *
*$##.##"
```


```
140 PRINT:PRINTUSING A$;C$(0);:
PRINTUSING B$;12.715265
150 PRINTUSING A$;C$(1);:PRINTU
SING B$;3.7363141
160 PRINTUSING A$;C$(2);:PRINTU
SING B$;10.35824221
170 PRINTUSING A$;C$(3);:PRINTU
SING B$;.5163733
180 PRINT @416,USING"LUCRO TOTA
L      $###,###.##";374241.5353
```


As linhas 10 a 90 exibem na tela o título do relatório, formatado de maneira simples com declarações **PRINT**. A linha 170 é particularmente interessante porque mostra como uma série de mensagens pode ser centrada na tela, usando-se **TAB**, que é calculado de acordo com o comprimento da mensagem.

A seção do programa que vai da linha 100 à linha 180 demonstra as características da declaração do **PRINT USING**. A linha 130 define **A$**, que pode ser usada para mostrar as primeiras seis letras das cadeias alfanuméricas definidas na linha 50. O comprimento da cadeia é o número de espaços entre sinais % (no MSX o sinal usado é a barra inversa), mais 2. Assim, o comprimento do cordão de saída levará em conta o espaço ocupado pelos sinais %.

B$ define o formato da saída numérica. A função principal da formatação numérica é alinhar uma coluna de números, padronizando-os para que tenham a mesma quantidade de algarismos antes e depois do ponto, que sejam tabulados tomando o ponto como referência, e assim por diante.

O número de dígitos a ser mostrado é definido pelo símbolo #. Inserindo-se o ponto decimal na cadeia de # define-se a sua posição. Examine a linha 130 do programa: **B$** serve para imprimir números com dois dígitos antes do ponto e dois dígitos depois.

Além disso, se o sinal $ for colocado à esquerda dos sinais #, ele será impresso nessa posição em todas as saídas, e será usado para indicar quantias monetárias.

O símbolo **antes dos sinais $ e # fará com que os espaços em branco à esquerda do campo definido para o valor numérico seja preenchido automaticamente por asteriscos. Este é um recurso usado no preenchimento de cheques. No campo de saída definido na linha 130 foram empregados quatro sinais diferentes. Outras combinações são possíveis.

A linha 140 imprimirá os seis primeiros caracteres de **C$(0)**, seguidos por oito espaços e um sinal de $. Note que a declaração **USING** utiliza a cadeia de formatação armazenada previamente em **B$**. O número a ser mostrado nesse exemplo será, portanto, arredondado, de modo a aparecer apenas com duas casas depois do ponto.

A linha 150 imprimirá os seis primeiros caracteres de A$, seguidos de *$3.74, pois sobrou um espapo em branco na formatação, que será preenchido por um asterisco. A linha 160 é similar à linha 140. Finalmente, a linha 170 imprimirá o número como *$0.52.

Se os números a imprimir forem muito grandes, pode-se recorrer a três tipos de formatação. A mais simples consiste na utilização de uma cadeia longa de sinais #. Por outro lado, para visualizá-la melhor, podemos empregar vírgulas para separar os dígitos em grupos de três (no Brasil usaríamos pontos, mas os interpretadores BASIC existentes em nossas máquinas seguem o padrão norte-americano). Isso foi feito na linha 180 do programa. A terceira alternativa é recorrer à forma exponencial. Para ver como ela funciona, substitua a linha 180 por:

```
180 PRINT @832,USING"LUCRO BRU-
TO     $##.##^^^^";374241.5353
```

Faça a modificação pertinente no programa para o MSX. Os quatro sinais de ↑ que aparecem no formato indicam o tamanho do campo do expoente. Os sinais # são interpretados, então, como o formato da mantissa.

A declaração **USING** pode ainda ser misturada com um **PRINT@** na mesma linha (computadores TRS-80 e TRS-Color) e várias **USING** podem ser colocadas na mesma linha. Isto dá grande flexibilidade na programação de telas.

Outro caractere de programação de formato usado com a **USING** e ao qual não nos referimos antes é o ponto de exclamação (!). Ele faz com que o programa imprima apenas o primeiro caractere de uma cadeia alfanumérica. Tente alterar a linha 130 do programa:

```
130 A$="  !":B$="
**$##.##"
```

Alguns programas utilizam a função **VAL** seguida de um número entre aspas, em vez de um número normal. Essa forma serve, muitas vezes, para economizar espaço. O computador normalmente recorre a bytes de memória para armazenar um número, o que o leva a desperdiçar memória em certas aplicações críticas. Assim, se você puser um número entre aspas e usar a função **VAL** para convertê-lo em seu valor numérico, poderá armazená-lo como se fosse um cordão. No caso de números curtos (ou seja, com menos de cinco dígitos), esse procedimento levará a uma economia de memória: um byte para cada dígito, um para cada aspa, e um para a função **VAL**.